UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Eder Matheus Silveira Félix

Uma Unidade de Medição Fasorial Sincronizada de Baixo Custo com Raspberry Pi 3

São Luís – MA
2018

Eder Matheus Silveira Félix

Uma Unidade de Medição Fasorial Sincronizada de Baixo Custo com Raspberry Pi 3

Monografia apresentada ao curso de Engenharia Elétrica da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Denivaldo Cicero Pavão Lopes.

São Luís –MA

2018

Félix, Eder Matheus Silveira.
Uma Unidade de Medição Fasorial Sincronizada de Baixo Custo com Raspberry Pi 3 / Eder Matheus Silveira Félix. - 2018.
102 p.

Orientador(a): Denivaldo Cicero Pavão Lopes.
Monografia (Graduação) - Curso de Engenharia Elétrica, Universidade Federal do Maranhão, São Luís, 2018.

1. GPS. 2. Medição Fasorial Sincronizada. 3. PMU. 4. Raspberry Pi. I. Lopes, Denivaldo Cicero Pavão. II. Título.

Eder Matheus Silveira Félix

Uma Unidade de Medição Fasorial Sincronizada de Baixo Custo com Raspberry Pi 3

Monografia apresentada ao curso de Engenharia Elétrica da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Aprovado em: 13 de julho de 2018.

BANCA EXAMINADORA

_____

Prof. Dr. Denivaldo Cicero Pavão Lopes (Orientador)

Departamento de Engenharia Elétrica - UFMA

_____

Prof. Msc. Antônio Dantas Maniçoba

Departamento de Engenharia Elétrica - UFMA

_____

Prof. Dr. Carlos Alberto Brandão Barbosa Leite

Departamento de Engenharia Elétrica - UFMA

A Deus

AGRADECIMENTOS

A Deus, pela vida, pelo discernimento, pela palavra e por todas as coisas que fez e tem feito em meu favor.

Aos meus pais, Luziana Sousa Silveira e Zaqueu Melo Félix, por todo o esforço para me sustentar, me educar e me preparar para a vida, pela confiança extraordinária que depositaram em mim e pelo companheirismo em todos os momentos.

A todos os meus familiares pelo apoio incondicional, em especial, a minha tia, Conceição Silveira e, seu esposo, Reginaldo Mendes, por terem aberto as portas da sua casa a mim, quando iniciei o curso de Engenharia Elétrica.

A todos os professores do Departamento de Engenharia Elétrica, pela experiência e pelos ensinamentos passados

Ao meu orientador, Prof. Denivaldo Lopes, pelo apoio, pelos ensinamentos e pela confiança durante o desenvolvimento deste trabalho.

A todos os amigos que fiz durante este curso, pelo companheirismo, respeito e amizade.

Enfim, aos meus irmãos da Comunidade Batista da Ilha, pelo apoio emocional e espiritual durante o desenvolvimento desta monografia.

“É melhor obter sabedoria do que ouro! É melhor obter entendimento do que prata!”

Provérbios 16, 16.

RESUMO

Este trabalho apresenta uma proposta de uma Unidade de Medição Fasorial Sincronizada – PMU (do inglês, *Phasor Measurement Unit*), utilizando a placa Raspberry Pi 3 como dispositivo de processamento. Uma PMU é um dispositivo que mede os fasores de tensões e correntes da rede elétrica. A medição é sincronizada através do pulso por segundo fornecido por um módulo receptor de GPS. A sincronização por GPS possibilita que os dados fasoriais medidos em lugares distantes geograficamente possuam a mesma referência de tempo, fornecendo uma visão global do estado da rede ao operador do sistema de potência. Um protótipo de uma PMU trifásica de baixo custo é proposto neste trabalho. O protótipo proposto faz a leitura das formas de onda de três sinais de corrente e de três sinais de tensão através de amostragem. Um algoritmo de Transformada Discreta de Fourier é implementado no Raspberry Pi 3 para gerar os fasores dos parâmetros medidos e os dados são enviados a um concentrador de dados fasoriais – PDC (do inglês, *Phasor Data Concentrator*). Circuitos de condicionamento de sinais com transformadores de potencial e transformadores de corrente são implementados. Testes de medição indicaram que o protótipo proposto realiza a leitura de três fases de tensão e corrente de maneira sincronizada e gera os dados fasoriais (módulo e fase) das grandezas medidas dez vezes por segundo. Os resultados obtidos são analisados com o auxílio do MATLAB.

Palavras-chave: Medição Fasorial Sincronizada. PMU. Raspberry Pi. GPS.

ABSTRACT

This work presents a proposal for a Synchronized Phasor Measurement Unit (PMU), using the Raspberry Pi 3 board as the processing device. The PMU is a device that measures voltage and current phasors of the electrical network. The measurement is synchronized through the pulse per second signal provided by a GPS receiver module. GPS synchronization enables measured phasor data in geographically distant locations to have the same time reference, providing an overview of the state of the network to the power system operator. A prototype of a low-cost three-phase PMU is proposed in this paper. The proposed prototype reads the waveforms of three current signals and three voltage signals through sampling. A Discrete Fourier Transform algorithm is implemented in the Raspberry Pi 3 to generate the phasors of the measured parameters and the obtained data is sent to a phasor data concentrator (PDC). Signal conditioning circuits with potential transformers and current transformers are implemented. Measurement tests indicated that the proposed prototype performs the three-phase voltage and current readings in a synchronized manner and generates the phasor data (modulus and phase) of the measured quantities ten times per second. The results obtained are analyzed with MATLAB.

Keywords: Synchronized Phasor Measurements. PMU. Raspberry Pi. GPS.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC	*Alternated Current*
A/D	Analógico/Digital
API	*Application Programming Interface*
DFT	*Discrete Fourier Transform*
FS	*Full-scale*
GPS	*Global Positioning System*
I2C	*Inter-Integrated Circuit*
IEEE	*Institute of Electrical and Eletronic Engineers*
IP	*Internet Protocol*
LAN	*Local Area Network*
NMEA	*National Marine Electronics Association*
PDC	*Phasor Data Concentrator*
PLL	*Phase-Locked Loop*
PMU	*Phasor Measurement Unit*
PPS	*Pulso por segundo*
SPI	*Serial Peripheral Interface*
SPMS	*Synchronized Phasor Measurement System*
TCP	*Transmission Control Protocol*
TC	Transformador de Corrente
TP	Transformador de Potencial
UART	*Universal Asynchronous Receiver-Transmitter*
UDP	*User Datagram Protocol*
UTC	*Coordinated Universal Time*
WAMS	*Wide Area Measurement System*

SUMÁRIO

1. INTRODUÇÃO

Os sistemas de energia elétrica têm passado nos últimos anos por uma evolução significativa, em especial, com a introdução de *SmartGrids*. A coleta de dados, processamento de dados e tomada de decisão baseado em dados e na geração de informações tem cada vez mais importância nas *SmartGrids*. Dentre os dados relevantes para uma rede de energia elétrica, destacam-se valores de tensão e corrente na forma de fasores, frequências e potências (ativas, reativas e aparentes).

Em redes de transmissão de energia elétrica de longas distâncias, faz-se necessário obter medidas ao longo da rede e, em específico, em suas extremidades. Entretanto, devido as longas distâncias, a obtenção de dados sobre a rede de energia elétrica em dado momento é complicada, pois há dificuldades na sincronização das leituras efetuadas ao longo da linha de transmissão, por exemplo, de fasores de tensão e corrente.

Dentre as soluções para leituras sincronizadas, tem-se utilizado o GPS para sincronizar relógios geograficamente distantes. Sendo assim, este trabalho de monografia apresenta a concepção, desenvolvimento e testes de um sistema de *software/hardware* que faz leituras sincronizadas de fasores de tensão e corrente. Desta forma, apresentamos uma solução que implementa uma PMU de baixo custo baseado em Raspberry Pi 3 e GPS.

1.1. Contexto Tecnológico

As redes elétricas tradicionais, provenientes do século passado, apresentam uma série de limitações frente à demanda cada vez maior dos consumidores e ao surgimento de fontes renováveis e distribuídas de energia. Segundo Monti e Ponti (2014), o sistema elétrico atual foi projetado sob a suposição de que a geração é controlável e previsível e, além disso, a geração é centralizada quanto o possível para facilitar a tarefa de agendamento do balanço de potência.

No entanto, o cenário atual apresenta o surgimento de políticas para diminuir o impacto ambiental através da criação e implantação de cada vez mais fontes renováveis de energia. O crescimento da geração descentralizada e de cargas não-lineares conectadas à rede leva a um aumento considerável da complexidade do sistema elétrico como um todo, tornando a tarefa de manter a qualidade de energia mais desafiadora.

A proposta mais recorrente e aceita no meio acadêmico para fazer frente ao aumento da complexidade do sistema elétrico são as *SmartGrids* (Redes Inteligentes, em português). As *SmartGrids* são consideradas a evolução do sistema elétrico tradicional, suportando fluxo de energia e informação em dois sentidos, criando uma rede elétrica automática e distribuída (FANG *et al.*, 2012).

Dentro do contexto das *SmartGrids*, existem os subsistemas de monitoramento, que são responsáveis por colher os dados da rede elétrica, tais como: *status* de equipamentos e linhas, consumo de energia em tempo real, grandezas elétricas (tensão, corrente, potência, frequência etc.) e o estado de operação do sistema como um todo. Umas das principais tecnologias empregadas no monitoramento inteligente da rede elétrica é a Unidade de Medição Fasorial Sincronizada – PMU.

A PMU é um dispositivo que realiza a leitura das formas de onda de tensão e corrente, fornecendo os parâmetros fasoriais (módulo e fase) destas grandezas de maneira sincronizada. Os dados obtidos por uma PMU possuem uma referência precisa de tempo, geralmente fornecida pelo Sistema de Posicionamento Global – GPS (do inglês, *Global Positioning System*). Desta forma, é possível, por exemplo, obter as diferenças de fase de tensões e correntes entre barramentos de subestações geograficamente distantes entre si.

1.2. Problemática

O problema abordado neste trabalho pode ser descrito através da seguinte questão: De que forma os fasores de tensão e corrente da rede elétrica podem ser medidos e quais dispositivos utilizar para esta aquisição, respeitando as interfaces disponíveis no Raspberry Pi 3 e atendendo aos requisitos de custo e desempenho?

1.3. Motivação

A existência de tecnologias baratas de prototipagem, como o Raspberry Pi 3, e a necessidade atual de fornecer soluções inovadoras para solucionar os problemas remanescentes das redes elétricas tradicionais são as principais motivações para a realização deste trabalho de pesquisa.

1.4. Solução proposta

A solução proposta neste trabalho pode ser dividida em *hardware* e *software*.

Em termos de *hardware*, é proposto um circuito de condicionamento de sinal com aplicação de transformadores de corrente – TC e transformadores de potencial – TP para medição dos sinais de tensão e corrente da rede elétrica através de amostragem realizada por um conversor A/D.

Um módulo GPS é utilizado para receber a referência de tempo para sincronização das leituras dos sinais da rede elétrica.

Em termos de *software*, o Raspberry Pi 3 é utilizado como dispositivo de processamento dos dados amostrados. Um algoritmo de Transformada Discreta de Fourier é implementado no Raspberry Pi 3 para estimar os fasores dos sinais de três fases de tensão e três fases de corrente amostrados da rede elétrica.

Os dados obtidos no Raspberry Pi 3 são enviados a um computador em rede local via *Ethernet* ou *WiFi*.

1.5. Justificativa

Com o crescimento da complexidade dos sistemas elétricos atuais, o monitoramento da rede elétrica em uma grande área geográfica tem se tornado uma abordagem fundamental para diminuir ou evitar a ocorrência de falhas na rede.

O modelo tradicional de rede possui limitações que geram problemas como: blecautes, falta de informação em tempo real, indisponibilidade de dados sincronizados e com referência de tempo, monitoramento limitado e falta de infraestrutura de comunicações de dados (ALI *et al.*, 2017; MONTI; PONTI, 2015).

O monitoramento fasorial sincronizado do sistema de potência permite ao operador do sistema a concentração de dados que podem ser utilizados para identificar anomalias, determinar a localização e o tempo exato da ocorrência de distúrbios e determinar o estado de funcionamento do sistema em tempo real. Tais características contribuem para melhorar a qualidade de energia e o desempenho do sistema como um todo.

O custo da tecnologia de PMU ainda consiste em um obstáculo para que governos e o setor privado passem a investir de maneira coordenada e consistente em sistemas de medição

fasorial sincronizada – SPMS (do inglês, *Synchronized Phasor Measurement Systems*). Uma pesquisa realizada pelo Departamento de Energia dos Estados Unidos (US DEP. OF ENERGY, 2014), revela que o custo relacionado ao projeto, *hardware*, infraestrutura e comissionamento de uma PMU varia de $40.000,00 a $180.000,00.

Desta forma, diversos trabalhos têm sido propostos na literatura (ALEIXO, 2018; ALI *et al.*, 2017; FERREIRA, 2017; LIMA, 2015) com o objetivo de fornecer tecnologias de baixo custo para o desenvolvimento de PMU.

O desenvolvimento do presente trabalho contribui para a academia e para a sociedade civil, no sentido de que descreve o uso de tecnologias relativamente baratas, como o Raspberry Pi 3, para o desenvolvimento e divulgação de tecnologias emergentes, como os SPMS; e descreve também uma metodologia de pesquisa e prototipagem que pode ser aplicada e melhorada em trabalhos futuros.

1.6. Objetivos

Os objetivos da realização deste trabalho estão divididos em objetivo geral e específicos, como segue.

1.6.1. Objetivo Geral

Conceber, projetar e testar um sistema de *hardware* e *software* para Medição Fasorial Sincronizada utilizando o Raspberry Pi 3.

1.6.2. Objetivos específicos

Os objetivos específicos são:

- Propor e testar um circuito para condicionamento e filtragem de três sinais de tensão e três sinais de corrente para serem lidos por um conversor A/D;
- Ler o sinal de pulso por segundo – PPS (do inglês, *Pulse Per Second*) e obter dados de tempo e data do módulo GPS para sincronização das leituras dos sinais;
- Elaborar um algoritmo em linguagem C para o Raspberry Pi 3 que gere os valores fasoriais (módulo e fase) da componente fundamental de 60 Hz de 6 sinais a partir de suas amostras;

- Fornecer um protótipo de uma PMU que faça a medição de fasores de tensão e corrente de um circuito trifásico de maneira sincronizada e que envie os dados para um servidor em rede local;
- Reconstituir as formas de onda de tensão e corrente no MATLAB a partir das informações obtidas pela PMU.

1.7. Metodologia de pesquisa

A metodologia de pesquisa utilizada neste trabalho é descrita como segue:

1) Pesquisa bibliográfica sobre os seguintes tópicos:
   a. Redes inteligentes;
   b. Monitoramento e medição inteligente de energia;
   c. Sistemas de Medição Fasorial Sincronizada – SPMS;
   d. Unidade de Medição Fasorial Sincronizada – PMU;
   e. Sistemas embarcados, incluindo aplicações com Raspberry Pi 3;
   f. Processamento digital de sinais;
   g. Linguagem de programação C;
   h. Linguagem de programação Python;
   i. Redes de computadores.

2) Testes de amostragem de sinais senoidais de 60 Hz com o Raspberry Pi 3 utilizando um conversor A/D;

3) Testes de processamento dos sinais amostrados utilizando a ferramenta MATLAB, para a escolha da frequência de amostragem e número de amostras ideais para gerar os fasores com boa precisão;

4) Desenvolvimento e teste no Raspberry Pi 3 de algoritmos em linguagem C para processamento de sinais amostrados;

5) Testes de interrupção externa no Raspberry Pi 3, sendo o sinal de interrupção externa o pulso por segundo fornecido pelo módulo GPS;

6) Testes de transferência de dados entre o Raspberry Pi 3 e um computador em rede local via protocolo UDP;

7) Desenvolvimento de um protótipo de uma PMU, e realização de testes de aquisição de sinais de uma fonte trifásica;

8) Análise qualitativa e quantitativa dos resultados obtidos.

1.8. Organização dos capítulos

O capítulo 2 apresenta uma revisão bibliográfica sobre as teorias utilizadas para o desenvolvimento do presente trabalho. Conceitos de sinais AC, fasores, conversão analógico-digital, processamento digital de sinais e computação paralela são apresentados.

O capítulo 3 apresenta uma revisão bibliográfica dos principais conceitos dos Sistemas de Medição Fasorial Sincronizada. As PMUs, PDCs e GPS são descritos e, por fim, a Norma IEEE C37.118 é apresentada.

O capítulo 4 apresenta a proposta de uma PMU de baixo custo com Raspberry Pi 3. A modelagem do sistema proposto é descrita em forma de diagramas. A metodologia utilizada para a aquisição de sinais da rede e para o processamento dos dados no Raspberry Pi 3 é apresentada.

O capítulo 5 apresenta o protótipo desenvolvido que implementa a modelagem descrita no capítulo 4. Os dispositivos utilizados, o tipo de montagem feita e o funcionamento do protótipo são descritos. Os testes realizados também são apresentados, assim como os principais resultados obtidos.

O capítulo 6 trata das conclusões, apresenta os objetivos alcançados, as limitações e os trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma revisão bibliográfica sobre as tecnologias utilizadas no desenvolvimento deste trabalho.

A representação de sinais elétricos no domínio do tempo e no domínio da frequência é apresentada. Conceitos sobre amostragem de sinais, conversão A/D e processamento digital de sinais com a Transformada Discreta de Fourier são descritos, assim como os principais conceitos envolvendo a programação paralela de computadores.

O Raspberry Pi 3 também é apresentado, abordando-se suas principais características e, por fim, uma breve explanação sobre protocolos de redes TCP/IP é fornecida.

2.1. Representação Fasorial de Sinais Elétricos AC

Um sinal elétrico em corrente alternada – AC (do inglês, *Alternated Current*) é representado matematicamente pela Equação 1 (PHADKE; THORP, 2008).

$$x(t) = X_m cos(2\pi f_o . t + \varphi) \qquad \text{(Eq. 1)}$$

Onde:

- X_m é a amplitude do sinal;

- f_o é a frequência fundamental do sinal em Hertz;

- φ é o ângulo de fase do sinal em radianos;

A Figura 1 mostra uma representação gráfica do sinal x(t), que é um sinal periódico de período $T_o = 1/f_o$ segundos. A fase φ de x(t) é o ângulo correspondente ao tempo decorrido entre os instantes t = 0 e $x(t) = X_m$. O valor RMS (do inglês, *Root Mean Square*) de x(t) é dado por $X_m/\sqrt{2}$.

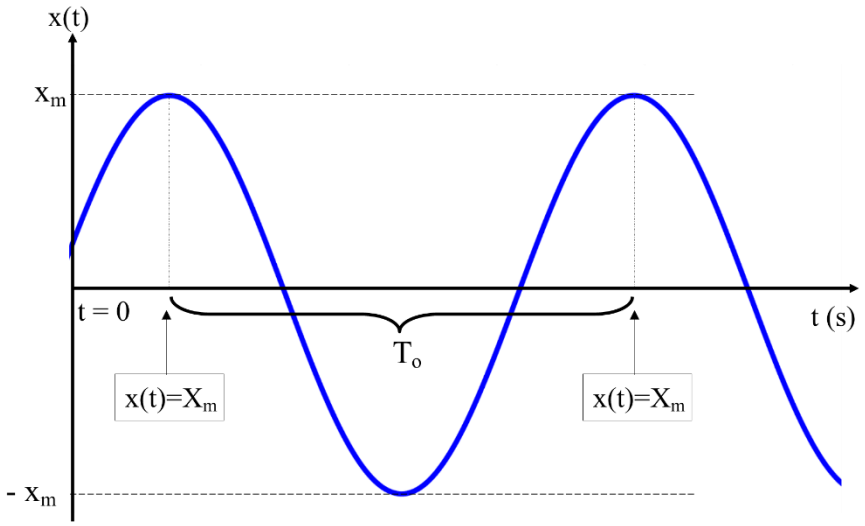

Figura 1. Representação gráfica de um sinal AC x(t).

O sinal dado pela Equação 1 pode ser representado por um número complexo X, chamado de **fasor**. A operação matemática que transforma uma função cosseno x(t) em um fasor X é chamada de Transformada Fasorial (NILSSON; RIEDEL, 2008). A definição do fasor é expressa pelas Equações 2 e 3 (PHADKE; THORP, 2008).

$$x(t) \leftrightarrow X = (X_m/\sqrt{2}) \times e^{j\varphi} \quad \text{(Eq. 2)}$$

$$X = (X_m/\sqrt{2}) \times [cos\varphi + jsen\varphi] \quad \text{(Eq. 3)}$$

A Equação 2 corresponde à forma polar do fasor. Na forma polar, é possível representar uma onda senoidal em termos do módulo (geralmente o valor RMS) e fase φ do sinal. A Equação 3 expressa a forma algébrica do fasor, que pode ser obtida diretamente da forma polar através da Equação de Euler.

Os fasores podem ser representados graficamente através dos diagramas fasoriais. A Figura 2 mostra o diagrama fasorial de X em comparação com a representação no domínio do tempo de x(t).

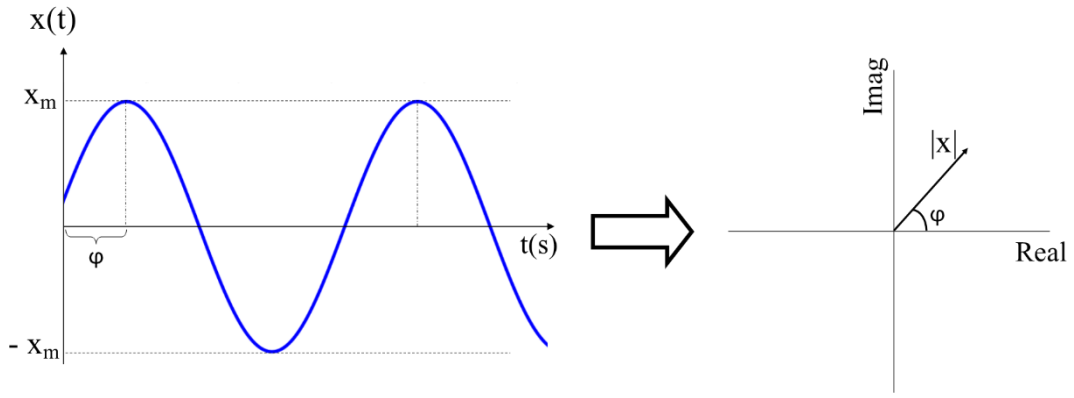

Figura 2. Diagrama fasorial de uma onda senoidal (Fonte: Baseado em PHADKE; THORP, 2008).

Os fasores podem representar, de maneira simplificada, **senóides puras**, ou seja, senóides que possuem apenas a componente fundamental de frequência. Geralmente, os sinais obtidos da rede elétrica possuem algum tipo de distorção harmônica, e para se obter a representação fasorial destes sinais, é necessário que a componente fundamental seja "extraída" do sinal. Isto pode ser feito através da Transformada de Fourier.

Em sistemas digitais, em que os sinais são amostrados, a Transformada Discreta de Fourier – DFT (do inglês, *Discrete Fourier Transform*) é utilizada. Esta técnica é detalhada na próxima seção.

2.2. Processamento Digital de Sinais

Os sinais provenientes da rede elétrica são analógicos, isto é, são de tempo contínuo e podem assumir infinitos valores diferentes. As ferramentas computacionais utilizadas para a análise destes sinais, no entanto, possuem capacidade limitada de armazenamento e processamento.

Os sinais analógicos necessitam passar por um processo de discretização para que possam ser armazenados e processados pelos computadores.

2.2.1. Amostragem, Conversão A/D e Quantização

Sinais analógicos são convertidos em sinais discretos por meio da operação de amostragem. A amostragem se dá pela obtenção da informação de amplitude do sinal analógico

em instantes de tempo específicos determinados pelo **intervalo** ou **período de amostragem T_s** (NALON, 2009). A Figura 3 demonstra a amostragem de um sinal x(t).

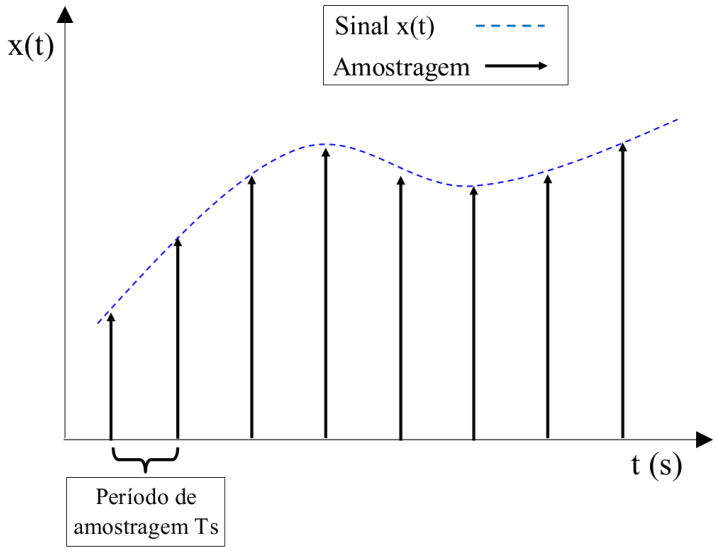

Figura 3. Amostragem de um sinal x(t) (Fonte: Baseado em NALON, 2009).

A informação resultante da amostragem é uma sequência x[n] que contém a amplitude do sinal x(t) em instantes específicos.

A amplitude de um sinal x(t), quando este é convertido para o formato digital, é dividida em níveis. Cada nível de amplitude é representado no mundo digital por um código binário. A conversão do sinal amostrado em um sinal codificado dá-se através dos processos de **quantização e codificação** (WILLISTON, 2009).

Quantização é o processo pelo qual qualquer valor contido em um dado intervalo de amplitude é convertido no valor central deste intervalo. A codificação é a representação dos níveis de quantização por números binários.

O conversor A/D é um dispositivo que realiza os processos de amostragem, quantização e codificação de um sinal analógico.

O fundo de escala – FS (do inglês, *full-scale*) é o intervalo total de amplitude que o conversor A/D suporta. A Tabela 1 mostra um exemplo de quantização e codificação. Considera-se um fundo de escala de 8 V, e uma faixa de representação binária de 3 bits.

Tabela 1. Quantização e codificação de um sinal em 3 bits.

Intervalo de amplitude (V)	Valor central	Código binário
0 <= x(t) < 0,5	0,0	000
0,5 <= x(t) < 1,5	1,0	001
1,5 <= x(t) < 2,5	2,0	010
2,5 <= x(t) < 3,5	3,0	011
3,5 <= x(t) < 4,5	4,0	100
4,5 <= x(t) < 5,5	5,0	101
5,5 <= x(t) < 6,5	6,0	110
6,5 <= x(t) < 7,5	7,0	111

A resolução do conversor A/D é o intervalo de amplitude compreendido entre os níveis de quantização, chamada de **quantização Q**, dada por:

$$Q = \frac{FS}{2^N} \qquad \text{(Eq. 4)}$$

Onde N é o número de bits utilizados pelo código numérico para representar o valor analógico convertido.

Um conversor A/D de N bits pode representar 2^N níveis diferentes do sinal de entrada. O conversor A/D do exemplo da Figura 4 é um conversor de N = 3 bits e FS = 8 V, logo a quantização deste conversor é Q = 1 V.

Observa-se que o conversor A/D não é capaz de representar o valor máximo do fundo de escala. O máximo valor representado por um conversor A/D é FS – Q. No caso da Tabela 1, o valor máximo representado pelo conversor A/D é 7 Volts.

2.2.2. *Aliasing* e o critério de Nyquist

O critério de Nyquist, também conhecido como o **teorema da amostragem**, afirma que a frequência de amostragem deve ser, no mínimo, duas vezes a maior componente de frequência do sinal a ser amostrado. Se a frequência de amostragem for menor que este limite, ocorrerá um fenômeno chamado *aliasing*.

O *aliasing* é um efeito causado pela subamostragem de um sinal. Quando um sinal é subamostrado, componentes indesejadas de baixa frequência aparecem no espectro de Fourier. A Figura 4 exemplifica a ocorrência de *aliasing*. Observa-se que, por conta da subamostragem

do sinal em azul, ocorre a amostragem de um sinal em uma frequência menor (sinal em preto tracejado).

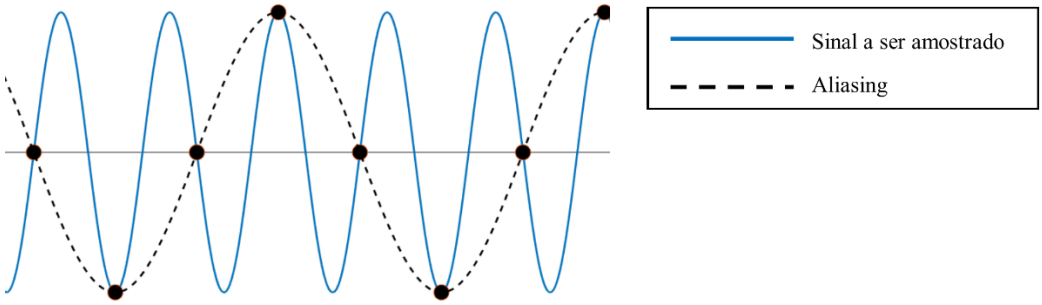

Figura 4. Exemplo de aliasing (Fonte: Baseado em NALON, 2009).

Mesmo que a frequência de amostragem seja definida para um valor que respeite o teorema de Nyquist, o sinal amostrado pode estar contaminado por componentes de alta frequência que irão aparecer no espectro em frequências mais baixas, causando sobreposição e perda de informações.

Para evitar este tipo de problema, propõe-se o uso de um filtro *anti-aliasing* na entrada do conversor A/D para garantir a atenuação de sinais com frequência superior à frequência de Nyquist. Para isso, um filtro passa-baixas com frequência de corte $0{,}5 \cdot f_s$ é utilizado, onde f_s é a frequência de amostragem (NALON, 2009).

2.2.3. A Transformada Discreta de Fourier

Em processamento digital de sinais, a representação de sinais no domínio da frequência fornece uma abordagem computacionalmente mais eficiente em termos de processamento, se comparada à representação no domínio do tempo (WILLISTON, 2009).

A Transformada Discreta de Fourier – DFT é uma ferramenta matemática que permite a representação de um sinal amostrado através de uma sequência numérica X[k], em que cada X[k] representa uma componente de frequência do sinal original x(t). A Equação 5 expressa a definição da DFT (NALON, 2009).

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \quad (k = 0,1,2,\dots,N-1) \qquad \text{(Eq. 5)}$$

Onde:

- x[n] são as amostras do sinal x(t);

- X[k] é a transformada discreta de Fourier;

- N é o número de amostras obtidas do sinal x(t).

O resultado da operação da DFT sobre x[n] é um conjunto de N números complexos. Cada número complexo representa uma senóide com frequência múltipla da frequência fundamental f = 1/NT$_S$, onde T$_S$ é o período de amostragem do sinal. A Equação 6 revela o conjunto de frequências computadas pela DFT (ROBERTS, 2003).

$$f = 0, \frac{1}{NT}, \frac{1}{NT} \times 2, \frac{1}{NT} \times 3, \ldots, \frac{1}{NT} \times k, \ldots, \frac{1}{NT}(N-1) \quad \text{(Eq. 6)}$$

Deve-se observar, porém, que a DFT é simétrica em relação ao ponto N/2, que corresponde à metade da frequência de amostragem f_s. Os números complexos compreendidos entre k = N/2 +1 e k = N-1 são números complexos conjugados daqueles compreendidos entre k = 1 e k = N/2 -1 (ROBERTS, 2003).

A DFT pode ser utilizada para se obter o módulo e ângulo de fase de uma determinada componente de frequência de um sinal x(t) (uma das funções de uma PMU).

Para processar um sinal de 60 Hz utilizando a DFT, calcula-se a frequência de amostragem através da Equação 7.

$$f_s = 60 \times (\text{nº de amostras por ciclo}) \quad \text{(Eq. 7)}$$

É necessário garantir que a janela de amostragem compreenda um número inteiro de ciclos do sinal de 60 Hz. Desta forma, o número N de amostras lidas do sinal deve ser definido pela Equação 8.

$$N = (\text{nº de ciclos amostrados}) \times (\text{nº de amostras por ciclo}) \quad \text{(Eq. 8)}$$

Cada número complexo obtido pela operação de DFT sobre um conjunto de amostras x[n] pode ser escrito como um sinal senoidal de frequência k/NT, dado pela Equação 9 (ROBERTS, 2003).

$$x_k(t) = \frac{2}{N}|X[k]|cos(2\pi(k/NT)t + ang(X[k])) \quad \text{(Eq. 9)}$$

Onde:

- $0 \leq k \leq N - 1$;

- |X[k]| é o módulo do número complexo X[k];

- ang(X[k]) é o ângulo do número complexo X[k] em radianos.

Desta forma, um sinal senoidal x(t) pode ser amostrado e processado em um sistema computacional e, posteriormente, pode ser recuperado através da Transformada Discreta de Fourier. O sinal da Equação 9 pode ser expresso na forma fasorial (ROBERTS, 2009):

$$\widehat{X_k} = \frac{2}{N}|X[k]|\angle ang(X[k]) \quad \text{(Eq. 10)}$$

A DFT é a principal ferramenta matemática utilizada neste trabalho para realizar o cálculo dos fasores.

2.3. Conceitos de computação paralela

Um programa de computador é definido por uma sequência de instruções sobre uma determinada região de memória à qual este programa tem acesso. O conjunto de instruções que forma um programa computacional pode ser executado de forma **serial** ou **paralela** (BARNEY, 2018).

- **Computação serial**
  - Um problema é divido em uma série de instruções;
  - As instruções são executadas uma após a outra, sequencialmente;
  - Utiliza apenas uma unidade de processamento;
  - Apenas uma instrução pode ser executada em um dado instante.

- **Computação paralela**
  - Um problema é quebrado em partes individuais que podem ser resolvidas concorrentemente;
  - Cada parte é quebrada em uma série de instruções;
  - As instruções de cada parte podem ser executadas simultaneamente em diferentes processadores;
  - Um mecanismo geral de controle e coordenação é empregado.

Computadores modernos utilizados no dia-a-dia como os *notebooks* e *smartphones* possuem arquiteturas computacionais com múltiplos processadores/núcleos. Para que a capacidade computacional destes dispositivos seja aproveitada com eficiência, se faz necessário o desenvolvimento de modelos e plataformas de *software* de computação paralela.

Um dos modelos de programação paralela mais utilizados é o *multithreading*, que consiste na execução de vários fios de execução (*threads*) concorrentes dentro de um mesmo programa. O modelo de *multithreading* é utilizado neste trabalho e é detalhado nas próximas subseções.

2.3.1. Processos e *Threads*

Lopes (2009) define **processo** como um programa computacional em curso de execução. Cada processo possui um conjunto de endereços de memória ao qual tem acesso e pode ler ou escrever informações.

Uma ***thread*** corresponde a uma linha de execução de um processo e um processo é formado por uma ou mais *threads*. A Figura 5 ilustra um processo formado por várias *threads*.

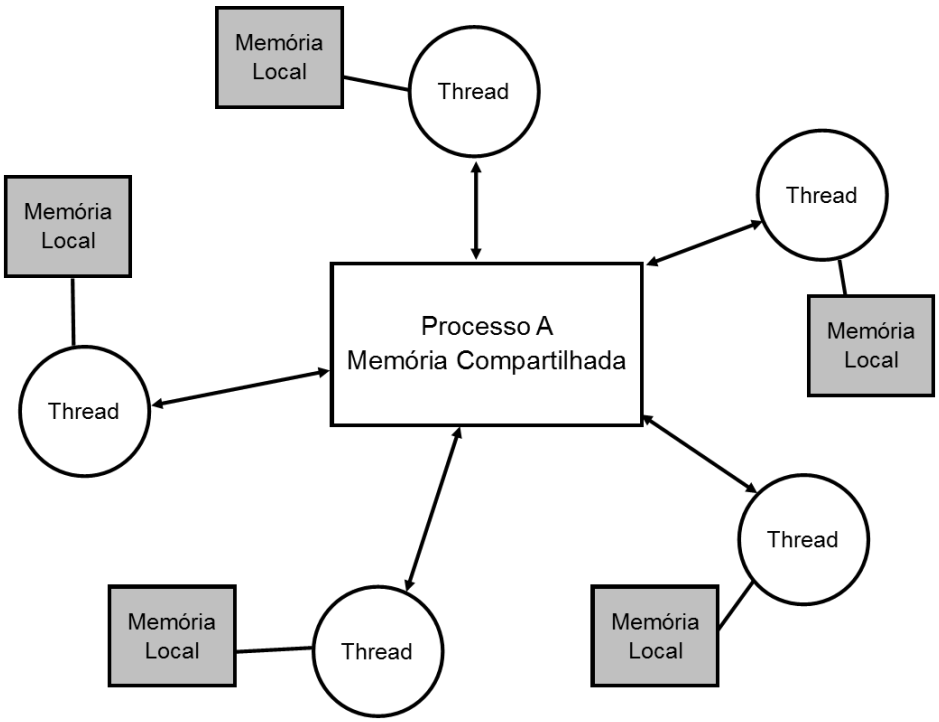

Figura 5. Processo computacional composto por várias threads (Adaptado de BARNEY, 2018).

Observa-se que cada *thread* possui seu próprio espaço de memória, que inclui variáveis locais, contador de programa, pilha etc. Uma *thread*, por si só, é um programa de computador. Porém, este programa compartilha recursos com outras *threads* de um determinado processo.

As *threads* podem ser executadas em paralelo (concorrentemente) ou não, a depender da aplicação.

2.3.2. Sincronização de *threads*

As *threads* pertencentes a um determinado processo podem comunicar-se entre si através do compartilhamento de locais de memória ou de variáveis globais (BARNEY, 2018).

No entanto, o acesso simultâneo de várias *threads* a uma determinada variável ou espaço de memória para operações de escrita e leitura pode tornar o conteúdo desta variável inconsistente.

A **região crítica** de um programa é definida como o trecho de código do programa que contém acesso à recursos compartilhados entre diferentes tarefas (*threads*) (LI; YAO, 2003; LOPES, 2009).

O acesso à região crítica deve ser coordenado através da sincronização das tarefas. Esta sincronização pode ser feita de duas formas: exclusão mútua e/ou semáforos.

Exclusão mútua

A exclusão mútua consiste na garantia de que apenas um processo/*thread* irá acessar a região crítica por vez. Isto é feito através das operações de bloqueio (*lock)* e desbloqueio (*unlock*) (LOPES, 2009). A Figura 6 mostra o fluxograma de um exemplo simples do uso de exclusão mútua.

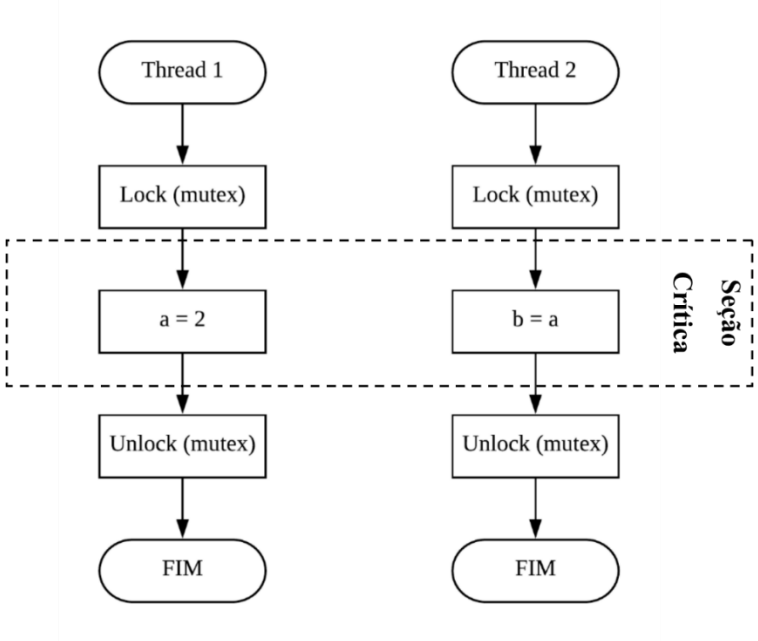

Figura 6. Exemplo de exclusão mútua (Adaptado de LOPES, 2009).

O comando *lock*, referente à variável de mutex (abreviação do termo *mutual exclusion* – exclusão mútua), bloqueia a execução de qualquer outra tarefa que encontre o comando *lock (mutex)* em seu código. O comando *unlock* desbloqueia o *mutex*.

A exclusão mútua auxilia na tarefa de manter os dados íntegros, pois evita a chamada *race condition*, que se refere à condição em que dois ou mais processos/*threads* tentam acessar o mesmo recurso ao mesmo tempo (GRAMA *et al.*, 2003).

Semáforos

Os semáforos são entidades implementadas por contadores que podem ser utilizadas para realizar a sincronização de tarefas/*threads* (LI, YAO, 2003; LOPES, 2009).

Existem duas operações básicas que definem os semáforos: P(s) e V(s). Onde *s* é um contador. A operação P(s) decrementa o valor *s*, mas assegura que *s* nunca será negativo e espera até que *s* seja positivo. A operação V(s) automaticamente incrementa o valor de *s* (LOPES, 2009).

Um semáforo iniciado com valor de $s = 1$ pode funcionar da mesma forma que um *mutex*. No algoritmo da Figura 5, basta substituir o comando *lock* por P(s) e substituir o comando *unlock* por V(s), por exemplo.

2.3.3. Modelo produtor-consumidor

O uso de semáforos é especialmente útil no caso em que existe uma tarefa que produz uma informação, e existe outra tarefa que consome esta informação. Com o uso dos semáforos, é possível implementar uma lógica na qual o consumidor da informação aguarda a produção desta informação por parte do produtor. Este modelo de sincronização é chamado de modelo **produtor-consumidor** (LOPES, 2009).

A Figura 7 mostra o fluxograma de um exemplo de modelo produtor-consumidor, no qual a *thread* 2 acessa a variável *a* somente após a *thread* 1 atualizar a variável *a*.

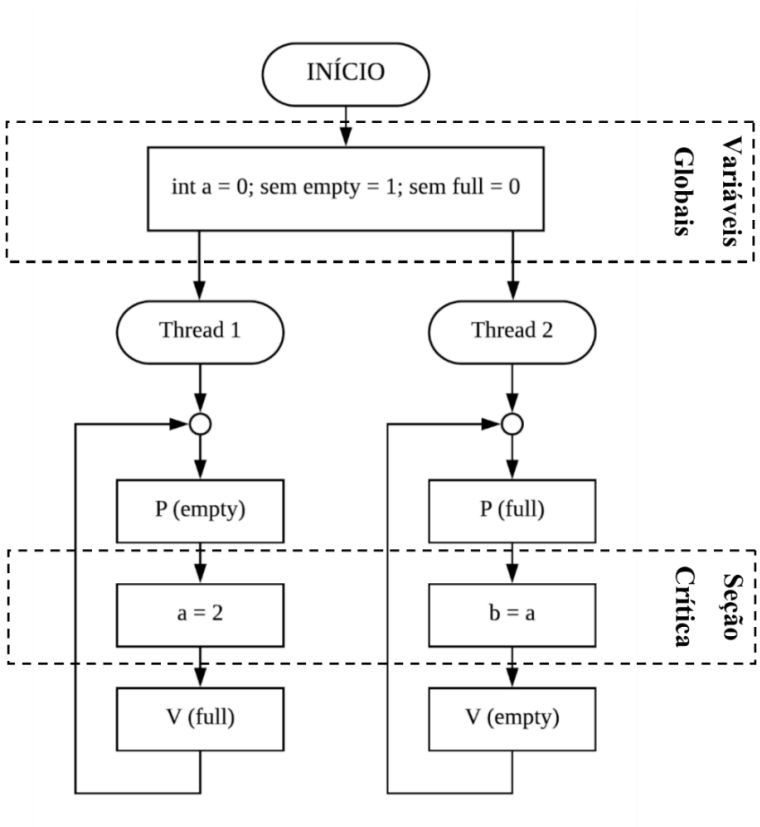

Figura 7. Exemplo de modelo produtor-consumidor (Adaptado de LOPES, 2009).

No exemplo da Figura 7, a *thread 2* espera até que a *thread 1* execute o comando *V(full)* para acessar a variável *a*. A *thread 1*, por sua vez, aguarda a execução do comando *V(empty)* na *thread 2* para que possa produzir novamente a informação. Neste tipo de aplicação, os semáforos podem assumir apenas dois valores: 0 ou 1. Neste caso, são chamados de **semáforos binários**.

A utilização de semáforos binários pode gerar atrasos indesejados, pelo fato dos processos terem que aguardar o incremento do semáforo para que possam atuar.

Os **semáforos contadores de recursos** são utilizados para evitar estes atrasos. Neste tipo de implementação, a informação a ser produzida e consumida é colocada em um *buffer* limitado de dados (LOPES, 2009). O fluxograma da Figura 8 exemplifica o uso de semáforos contadores com *buffer* limitado.

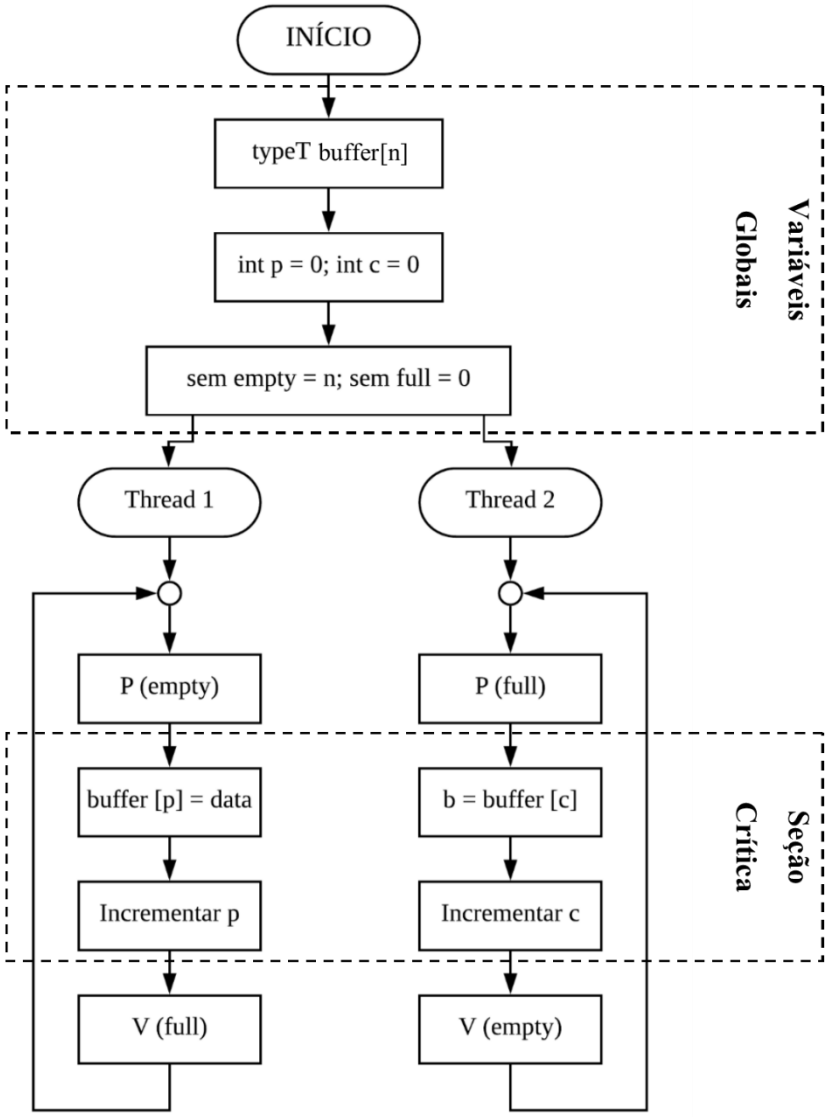

Figura 8. Utilização de semáforos contadores de recursos limitados (Adaptado de LOPES, 2009).

Com o decorrer da execução do algoritmo da Figura 8, o produtor vai preenchendo o *buffer* e decrementando o semáforo *empty* (que é do mesmo tamanho do *buffer*). O consumidor, por sua vez, esvazia o *buffer*, incrementa o semáforo *empty* e decrementa o semáforo *full*.

Com este tipo de implementação, a *thread 1* não necessita esperar a leitura dos dados por parte da *thread 2*. O produtor (*thread 1*) pode produzir várias informações até encher o *buffer*. O consumidor (*thread 2*), uma vez que o *buffer* contenha vários elementos, pode realizar várias leituras sem esperar pela *thread 1*.

2.4. Raspberry Pi 3

O Raspberry Pi 3 é um minicomputador embarcado em uma placa do tamanho de um cartão de crédito que custa, no momento em que este documento é escrito, cerca de R$ 200,00 em lojas virtuais como americanas.com e filipeflop.com.

O Raspberry Pi 3 possui todos os componentes de um computador comum: processador, memória principal, memória secundária, interfaces de rede (*Ethernet* e *WiFi*), portas USB, saída HDMI e sistema operacional. O Raspberry Pi 3 possui também interfaces GPIO (do inglês, *General-Purporse Input and Output*) que permitem a comunicação com outros dispositivos via protocolos seriais I2C, UART e SPI. A Figura 9 mostra o Raspberry Pi 3 Model B, modelo utilizado neste trabalho.

Figura 9. Raspberry Pi 3 Model B (Fonte: RASPBERRY PI FOUNDATION, 2018).

O Raspberry Pi começou a ser desenvolvido na Universidade de Cambridge, Reino Unido, em 2006 (DENNIS, 2016). A ideia do projeto foi construir um dispositivo acessível e barato para o ensino de programação e arquitetura de computadores em escolas. A primeira versão do Raspberry Pi, o Raspberry Pi Model A, foi lançado em 2012, sendo comercializado a $ 25,00 (vinte e cinco dólares).

O baixo custo, e o alto poder de processamento fez com que o Raspberry Pi se tornasse bastante popular no Reino Unido e em muitos países do mundo. O incentivo e o *feedback* da comunidade desenvolvedora fizeram com que outras versões do Raspberry Pi fossem lançadas, trazendo melhorias à plataforma.

O Raspberry Pi 3 Model B possui as seguintes especificações (RASPBERRY PI FOUNDATION, 2018):

- CPU Quad Core Broadcom BCM2837 de 64 bits e 1,2 GHz;
- 1 GB de memória RAM;
- Internet *wifi* e *Bluetooth onboard*;
- Interface de rede *Ethernet*;
- 40 pinos GPIO;
- 4 portas USB (do inglês, *Universal Serial Bus*);
- Saída de vídeo composto e saída de áudio estéreo;
- Saída HDMI;
- Porta CSI para conexão de câmera;
- Porta DSI para conexão de *display touchscreen*;
- Entrada micro SD para carregar o sistema operacional e armazenamento de dados;
- Alimentação via entrada Micro USB de 5 V (requer fonte de 2,5 A);

Processador, memória RAM e o processador gráfico são encapsulados em um único chip, ou seja, um SoC (do inglês, *System on a Chip*). O processador utilizado no SoC Broadcom BCM2837 é um processador de arquitetura ARM com 4 núcleos físicos.

2.4.1. Sistema Operacional Raspbian

O sistema operacional utilizado no Raspberry Pi 3 para a realização deste trabalho é o Raspbian, uma distribuição Linux baseada no Debian. O Raspbian possui as bibliotecas padrão da linguagem C para sistemas Linux, incluindo APIs para criação e sincronização de *threads* (*pthread.h, semaphore.h*), criação de sockets para comunicação TCP e UDP (*arpa/inet.h, sys/socket.h*), comunicação serial SPI (*linux/spi/spidev.h, sys/ioctl.h*), entre outras (GAY, 2014).

O Raspbian pode ser utilizado no modo gráfico, como mostra a Figura 10. O modo gráfico oferece ao usuário uma interface amigável e simples de usar, contendo aplicativos de edição de texto, de escritório (LibreOffice), navegação na Web e IDEs (do inglês, *Integrated Development Environment* – Ambiente Integrado de Desenvolvimento) para desenvolvimento de programas em linguagens de programação como Python e Java.

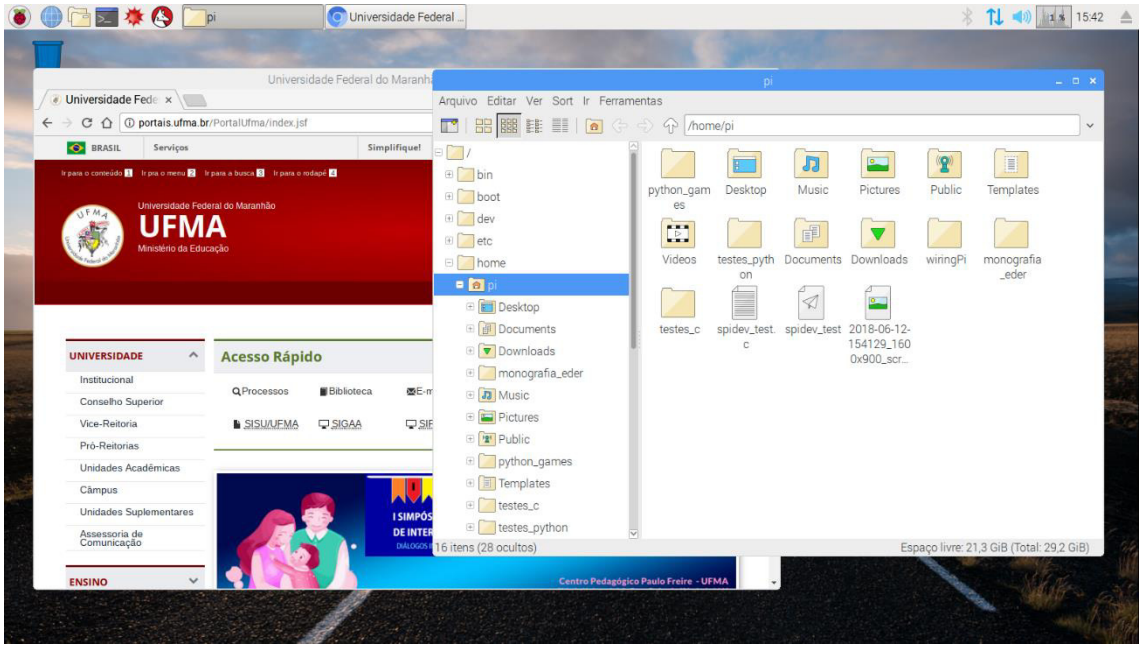

Figura 10. Modo gráfico do sistema operacional Raspbian.

O Raspbian oferece também a alternativa de se utilizar o sistema no modo texto, que consiste no terminal *shell* do Linux. A Figura 11 mostra a tela do Raspbian configurado no modo texto. O terminal do Linux oferece ferramentas avançadas para configuração do sistema, instalação ou remoção de programas e *drivers*, e desenvolvimento de programas em diversas linguagens de programação.

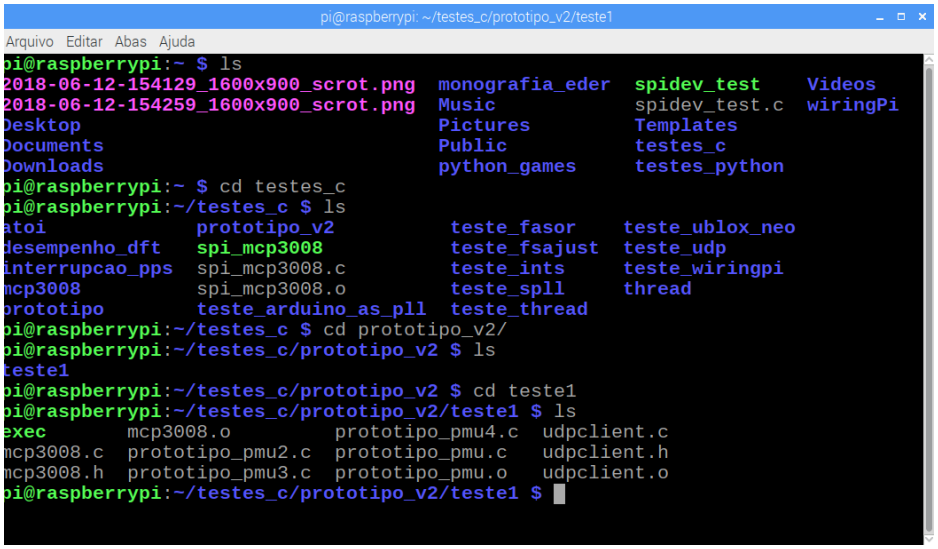

Figura 11. Modo texto (terminal do Linux) do sistema operacional Raspbian.

O modo texto é preferível em muitas aplicações, pois reduz consideravelmente o consumo de recurso computacional se comparado ao modo gráfico.

O compilador de linguagem C GNU GCC (FREE SOFTWARE FOUNDATION, 2018) está presente no Raspbian, e todas as suas funcionalidades podem ser acessadas através de comandos no terminal. Além de C, o GCC é capaz de compilar outras linguagens de programação como C++, Fortran, Objective-C, Objective-C++, Ada, Go e BRIG (STALLMAN, 2003).

2.4.2. Interface GPIO

O Raspberry Pi 3 Model B possui 40 pinos GPIO de uso geral (RASPBERRY PI FOUNDATION, 2018). Os pinos GPIO são interfaces utilizadas para conectar o Raspberry Pi ao mundo real. A Figura 12 indica a localização da pinagem GPIO na placa do Raspberry Pi 3.

Figura 12. Pinagem GPIO no Raspberry Pi 3 Model B (Fonte: RASPBERRY PI FOUNDATION, 2018).

Os pinos GPIO são interfaces elétricas digitais que operam com níveis de tensão de 0,0 V a 3,3 V e suportam a comunicação com outros dispositivos através de protocolos de comunicação serial. Os protocolos I2C, SPI e UART são suportados.

Os pinos da interface GPIO podem ser utilizados também como entradas de interrupção externa e modulação por largura de pulso – PWM (do inglês, *Pulse Width Modulation*).

Existem diferentes numerações para os pinos GPIO do Raspberry Pi 3. O desenvolvedor de aplicações para o Raspberry Pi deve estar atento ao tipo de numeração utilizada pela API com a qual se está desenvolvendo. A Tabela 2 mostra as numerações do GPIO *header* do Raspberry Pi 3 Model B.

Tabela 2. Numeração dos pinos GPIO do Raspberry Pi 3 Model B (Adaptado de THE PI4J PROJECT, 2018).

Função alternativa	NOME	*Pino #*		NOME	Função Alternativa
DC Power	3.3 V	1	2	5 V	DC Power
I2C - SDA1	GPIO**02**	3	4	5 V	DC Power
I2C - SCL1	GPIO**03**	5	6	Ground	
GPIO - GCLK	GPIO**04**	7	8	GPIO**14**	UART - TX
	Ground	9	10	GPIO**15**	UART - RX
GPIO_GEN0	GPIO**17**	11	12	GPIO**18**	GPIO_GEN1
GPIO_GEN2	GPIO**27**	13	14	Ground	
GPIO_GEN3	GPIO**22**	15	16	GPIO**23**	GPIO_GEN4
DC Power	3.3 V	17	18	GPIO**24**	GPIO_GEN5
SPI - MOSI	GPIO**10**	19	20	Ground	
SPI - MISO	GPIO**09**	21	22	GPIO**25**	GPIO_GEN6
SPI - CLK	GPIO**11**	23	24	GPIO**08**	SPI - CE0
	Ground	25	26	GPIO**07**	SPI - CE1
I2C ID EEPROM	ID_SD	27	28	ID_SC	I2C ID EEPROM
	GPIO**05**	29	30	Ground	
	GPIO**06**	31	32	GPIO**12**	
	GPIO**13**	33	34	Ground	
	GPIO**19**	35	36	GPIO**16**	
	GPIO**26**	37	38	GPIO**20**	
	Ground	39	40	GPIO**21**	

A numeração de 1 a 40 corresponde aos números dos pinos implementados fisicamente na placa do Raspberry Pi 3. A numeração na coluna NOME que vai de GPIO01 a GPIO26 corresponde à numeração dos pinos no SoC Broadcom BMC2837.

Ao contrário da numeração dos pinos físicos do Raspberry Pi, a numeração dos pinos do SoC não costuma mudar entre as versões do Raspberry Pi.

wiringPi

O *WiringPi* é uma biblioteca escrita em linguagem C que fornece interface de programação em C para o GPIO do Raspberry Pi (WIRING PI, 2018).

O *WiringPi* inclui o utilitário em linha de comando ***gpio*** que pode ser utilizado para programar e configurar os pinos GPIO. Além disso, o *wiringPi* fornece módulos em linguagem C que elevam o nível de abstração de funções primitivas que suportam:

- interface com dispositivos via protocolos de comunicação serial como SPI, I2C e UART;

- leitura e o acionamento dos pinos GPIO;
- interrupção externa nos pinos GPIO.

O *wiringPi* é utilizado neste trabalho para configurar interrupções externas. A comunicação via SPI e UART é feita através do uso de funções primitivas.

2.5. Protocolo de comunicação TCP/IP

O principal tipo de comunicação entre computadores utilizada atualmente é a Internet. O funcionamento da Internet consiste na definição de protocolos para envio e recebimento de pacotes de dados entre processos que estão em execução em computadores (ou *hosts*).

O conjunto de protocolos que permitem o funcionamento da Internet é chamado de **pilha TCP/IP**, que divide os protocolos de rede de computadores em quatro camadas: aplicação, transporte, internet e rede (TANENBAUM; WETHERALL, 2010).

Camada de aplicação

A camada de aplicação contém os protocolos que são executados nos sistemas finais e que definem o tipo de aplicação que está sendo executada. Alguns exemplos bem comuns são: HTTP (páginas Web), SMTP (serviço de email), DNS (serviço de nomes), entre outros.

A camada de aplicação utiliza os serviços da camada de transporte para realizar a transferência de seus pacotes.

Camada de transporte

A camada de transporte é responsável por realizar a transferência das mensagens da aplicação do remetente ao destinatário. Existem dois protocolos utilizados para transferência de dados na camada de transporte: o **TCP** e o **UDP**.

O TCP (do inglês, *Transmission Control Protocol*) é um protocolo da camada de transporte orientado à conexão, ou seja, requer que os *hosts* abram uma conexão para então iniciar a transferência dos dados. O TCP implementa controle de fluxo, controle de congestionamento e, além disso, garante a entrega das mensagens de forma íntegra e em ordem.

O UDP (do inglês, *User Datagram Protocol*) é um protocolo não orientado à conexão, ou seja, os *hosts* podem realizar a transferência de dados sem a necessidade de uma conexão previamente estabelecida. Ao contrário do TCP, o UDP não possui controle de fluxo e

congestionamento, não garante a entrega das mensagens e não garante a ordem da entrega das mensagens.

Desta forma, o UDP é um protocolo mais rápido e menos complexo que o TCP. No entanto, o TCP é mais seguro e garante a transferência de dados de forma confiável.

Quando pretende-se criar uma aplicação para a Internet, deve-se escolher se a aplicação realizará a transferência de dados via UDP ou TCP. Geralmente, não é necessário programar ou implementar o protocolo da camada de transporte a ser utilizado, pois o sistema operacional se encarrega desta tarefa. Basta o desenvolvedor definir em sua aplicação qual protocolo utilizar.

Camada de Internet

A camada de internet é responsável por realizar a entrega dos pacotes, não importando para esta camada o conteúdo deste pacote ou o protocolo da camada de transporte utilizado.

Os pacotes são transportados na camada de internet independentemente entre si e podem, inclusive, chegar fora de ordem ao destino, cabendo às camadas superiores (transporte e aplicação) reordená-los.

Os principais protocolos da camada de internet são o IP (do inglês, *Internet Protocol*) e o ICMP (do inglês, *Internet Control Protocol*).

Os protocolos da camada de Internet são executados nos sistemas finais e nos dispositivos do núcleo da rede, como roteadores e *switches*.

Camada de acesso à Rede

Esta é a camada de mais baixo nível no modelo TCP/IP, e corresponde aos protocolos e padrões utilizados para conectar os sistemas finais aos *links* que fornecem o serviço de internet. *Ethernet* e *WiFi* são exemplos de padrões definidos na camada de acesso à rede.

Os padrões e protocolos definidos nesta camada geralmente são implementados através do *hardware* dos dispositivos. O Raspberry Pi 3, por exemplo, possui *hardware* compatível com as tecnologias *Ethernet* e *WiFi*.

2.6. Síntese

Os principais pontos a destacar deste capítulo são listados como segue:

- Uma senóide pura pode ser representada por um número complexo, denominado **fasor**. O fasor contém as informações de magnitude (valor de pico ou valor RMS) e fase de um sinal senoidal;
- Sinais analógicos passam pelos processos de amostragem, quantização e codificação, para que possam ser entendidos e processados por computadores;
- A Transformada Discreta de Fourier é uma ferramenta matemática utilizada para recuperar as informações de uma senóide a partir de suas amostras;
- O desenvolvimento de programas computacionais em várias *threads* (*multithreading*) permite a utilização de recursos computacionais com mais eficiência, especialmente no caso de processadores constituídos de vários núcleos, como é o caso do Raspberry Pi 3;
- O *multithreading* requer a utilização de semáforos para sincronizar o acesso de várias tarefas simultâneas a recursos compartilhados, como as variáveis globais;
- O Raspberry Pi 3 Model B consiste em um computador embarcado em uma pequena placa de prototipagem, que possui um processador ARM v8 de 4 núcleos e 1 GB de memória RAM;
- O Raspberry Pi 3 possui sistema operacional Raspbian, baseado em Linux, que permite o desenvolvimento de projetos em várias linguagens de programação como C, C++, Python e Java;
- Existem dois protocolos da camada de transporte do modelo TCP/IP: o TCP e o UDP. O TCP garante segurança de transmissão, e o UDP garante maior velocidade de comunicação.

3. SISTEMAS DE MEDIÇÃO FASORIAL SINCRONIZADA – SPMS

No presente capítulo, o Sistema de Medição Fasorial Sincronizada – SPMS é apresentado. O contexto histórico e uma visão geral dos SPMS são discutidos, e as partes que compõem o sistema, como o GPS, a PMU e o PDC, são apresentadas, assim como a Norma IEEE C37.118, que regulamenta o funcionamento e o desempenho das PMUs.

3.1. Contexto Histórico

Estudos preliminares sobre relés computacionais datam do início dos anos 60 (THORP; PHADKE, 1991), quando os sistemas computacionais passaram a ser utilizados para realizar cálculos de curto-circuito e o processamento de dados de subestações.

Com a evolução dos computadores e o surgimento dos sistemas microprocessados, nas décadas de 1970 e 1980, várias pesquisas foram conduzidas no sentido de fornecer algoritmos e metodologias com o objetivo de tornar viável o cálculo de fluxo de potência, estimação de estado e proteção de sistemas de energia elétrica através do uso de computadores digitais. Segundo Phadke e Thorp (2008), a era moderna da medição fasorial surge com pesquisas na área de relés computacionais de linhas de transmissão.

Sabe-se, da teoria de análise de sistemas de energia elétrica, que o fluxo de potência ativa em uma linha de transmissão é diretamente proporcional ao seno da diferença de ângulo de fase (defasagem) entre as tensões das extremidades da linha, considerando um modelo simplificado (considera-se apenas as reatâncias) de linha de transmissão (MONTICELLI, 1983). Estimar esta diferença de fase é de fundamental importância para a estimação de estado e, consequentemente, para a proteção do sistema. A partir desta motivação e com o aumento da disponibilidade de ferramentas computacionais, os primeiros estudos sobre os SPMS surgiram no início dos anos 1980.

Na mesma época, o Sistema de Posicionamento Global (GPS) estava em implementação, e esta tecnologia passou a ser utilizada para sincronizar a medição fasorial em sistemas de medição de grandes distâncias – WAMS (do inglês, *Wide Area Measurement Systems*) (PHADKE; THORP, 2008). O surgimento do GPS marcou o início da "era moderna" dos SPMS, uma vez que o GPS fornece sincronia de tempo com maior precisão em relação a

outros sistemas da época, como o LORAN-C e GOES, via satélite, e o HBG, via transmissão de rádio.

O primeiro protótipo de uma PMU foi desenvolvido em 1988, nos laboratórios do Instituto Politécnico e Universidade Estadual da Virgínia (Virginia Tech), nos Estados Unidos (ZHANG *et al*, 2010).

Desde então, a tecnologia de SPMS tem se tornado a melhor alternativa para a implementação de WAMS. O sistema tradicional de medição de energia em grandes áreas, o SCADA (do inglês, *Supervisory Control And Data Acquisition*), embora apresente bom desempenho (ALEIXO, 2018), possui algumas limitações, como citam Anisha, Prasanna e Suyampulingam (2015): não possui sincronização por tempo; fornece apenas a magnitude dos sinais medidos; e possui baixa taxa de atualização, cerca de uma atualização a cada 2 ou 4 segundos.

3.2. Visão Geral dos SPMS

O Sistema de Medição Fasorial Sincronizada possui três componentes principais: O GPS, que fornece sincronização para a leitura dos sinais através do pulso por segundo (PPS) e fornece também a referência de horário e data; a PMU, que realiza a aquisição dos sinais da rede elétrica, faz o processamento destes sinais, e envia a informação obtida para o concentrador de dados fasoriais (PDC), que consiste no último componente. Cada um destes componentes será descrito com mais detalhes nas próximas subseções.

A Figura 13 mostra uma representação de um SPMS aplicado a um sistema de energia elétrica simples contendo geração, transmissão e distribuição.

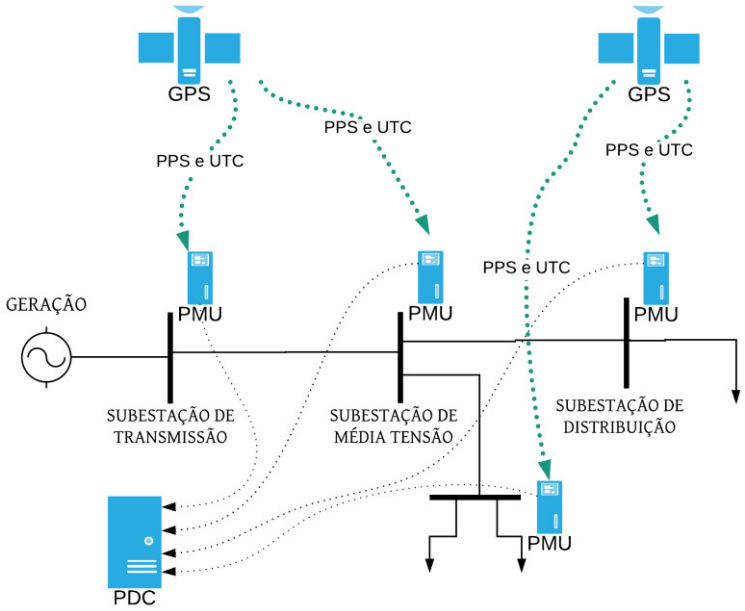

Figura 13. Modelo simples de um Sistema de Medição Fasorial Sincronizada (Fonte: Baseado em PHADKE; THORP, 2008).

O GPS fornece o PPS e o horário UTC para a PMU, que utiliza estas informações para sincronizar e etiquetar as medições de tensão e corrente. Como todas as PMUs estão operando sob o mesmo sinal de GPS, elas podem realizar a aquisição de dados da rede elétrica de forma sincronizada. Toda a informação gerada nas PMUs é então enviada ao PDC, que consiste em um servidor para o armazenamento de dados fasoriais.

3.3. Sistema de Posicionamento Global – GPS

O Sistema de Posicionamento Global – GPS (do inglês, *Global Positioning System*) é um sistema de navegação em escala global criado em 1978 pelo Departamento de Defesa dos Estados Unidos para fornecer informações de localização, navegação e de tempo para uso militar, sendo declarado em plena operação no ano de 1995 (VAIL *et al*, 2015).

O GPS conta com uma constelação de mais de 24 satélites divididos em 6 órbitas. Estes satélites estão posicionados de tal forma que, na maior parte do planeta, pelo menos seis satélites são visíveis. Frequentemente, até dez satélites são visíveis (PHADKE; THORP, 2008). A Figura 14 ilustra a órbita dos satélites do GPS.

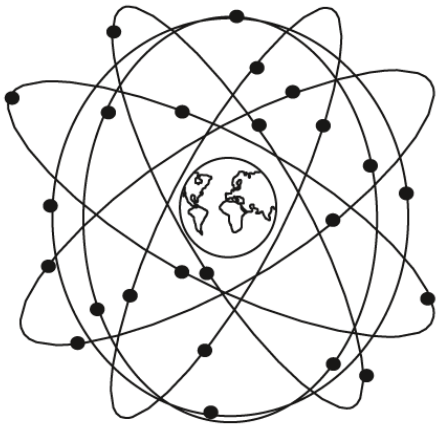

Figura 14. Conjunto de satélites que compõem o GPS (Fonte: PHADKE; THORP, 2008).

O GPS é comumente utilizado para localização e navegação de veículos terrestres, aviões e embarcações, além de dispositivos portáteis como celulares e *smartphones*.

Para aplicações em PMUs, necessita-se apenas das informações de data e horário e do pulso por segundo (PPS) fornecidos pelo GPS.

O PPS é enviado aos receptores de GPS a cada mudança de segundo no horário UTC. O UTC é a base de tempo legal adotada em todo o mundo. A Figura 15 mostra a forma de onda do pulso por segundo (PPS) obtido de um receptor GPS.

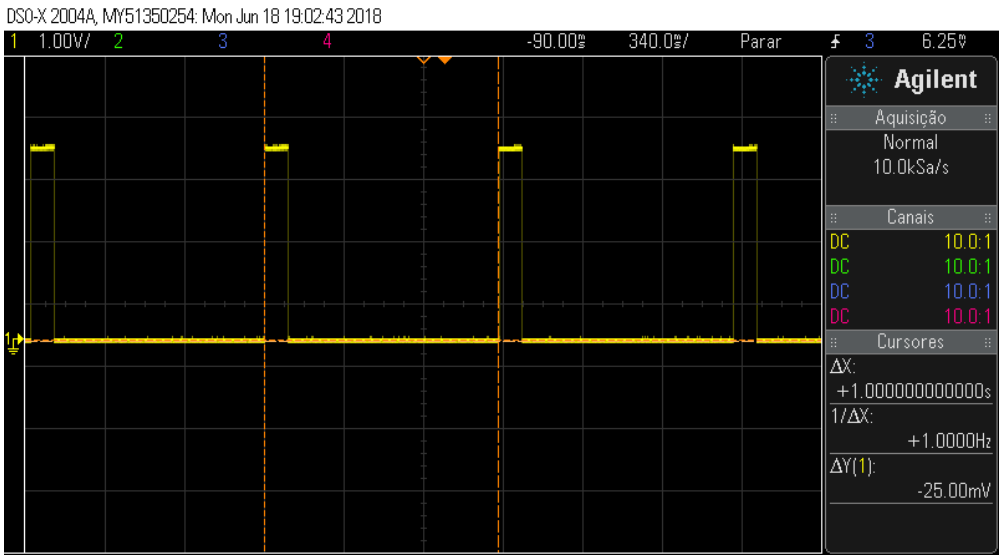

Figura 15. Forma de onda do PPS do GPS (Fonte: Autor).

O PPS é um sinal de alta precisão. Por definição, o erro máximo do PPS é de 1 µs (PHADKE; THORP, 2008). Sendo que, na prática, o erro raramente chega próximo desse valor.

Os dados enviados pelos satélites aos receptores de GPS seguem o padrão NMEA 0183. O padrão NMEA 0183 define o formato das sentenças que contém diversas informações fornecidas pelos satélites do GPS. Tais informações (referentes ao receptor GPS) incluem localização geográfica, velocidade horizontal em relação ao solo, velocidade vertical, horário, data, entre outras informações (NMEA, 2018).

A sentença RMC (do inglês, *Recommended Minimum Data for GPS*) contém as informações de data e horário UTC. Estas informações são necessárias para fornecer a etiqueta de tempo às leituras da PMU.

A sentença RMC é identificada através da sequência de caracteres "$GPRMC" que precede as informações a serem colhidas. Segue abaixo um exemplo de sentença RMC, colhida de um receptor GPS:

```
$GPRMC,184011.00,A,0233.54277,S,04418.46811,W,0.053,,180518,,,
A*73
```

As informações de data e horário UTC são retiradas da sentença supracitada como segue:

- "184011" é o horário UTC correspondente às 18 horas, 40 minutos e 11 segundos.
- "180518" é a data, que corresponde a 18 de maio (05) de 2018 (18).

Estas informações podem ser obtidas em software através da comparação de *strings*.

3.4. Sincrofasores

Como já discutido, um fasor é um número complexo que representa a componente de frequência fundamental de um sinal elétrico senoidal. O fasor pode ser entendido como uma "fotografia" do sinal elétrico, contendo o módulo e fase deste sinal.

O sincrofasor pode ser definido como um fasor com referência de tempo. Ou seja, o sincrofasor contém informações de módulo, fase e o tempo em que estas informações foram obtidas (ALI *et al*, 2017; IEEE, 2011a; LIMA, 2014).

Desta forma, fasores obtidos com a mesma referência de tempo estão sincronizados. Daí o cunho “sincrofasores”.

A sincronia da medição fasorial é de fundamental importância para a obtenção da defasagem entre sinais elétricos. A Figura 16 ilustra a definição de sincrofasor. O sinal de PPS é utilizado como sinal de sincronia.

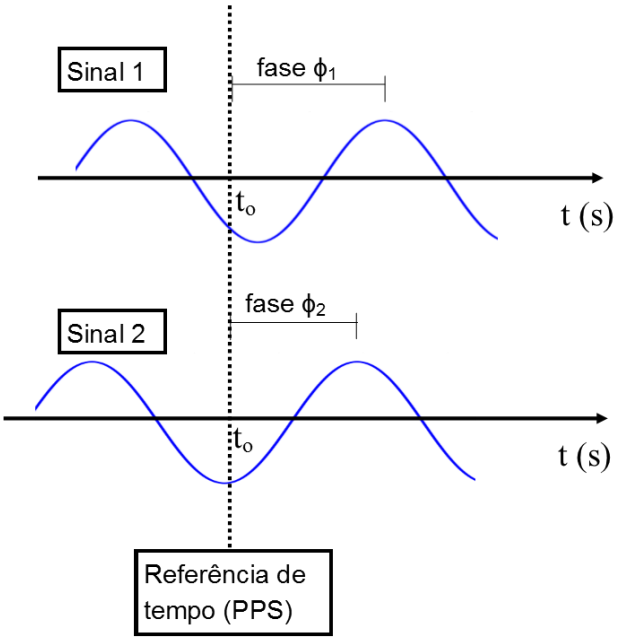

Figura 16. Esquema que representa dois sincrofasores (Fonte: Baseado em IEEE, 2011a).

A ocorrência do PPS é o instante inicial da medição fasorial, que consiste no tempo tomado como referência para a medição do ângulo de fase.

As informações fasoriais dos sinais 1 e 2 (módulo e fase) da Figura 16, juntamente com a referência de tempo t_o, comum aos dois sinais, formam os sincrofasores 1 e 2, respectivamente.

3.5. Unidade de Medição Fasorial Sincronizada – PMU

A Unidade de Medição Fasorial Sincronizada – PMU é um dispositivo que faz a leitura de sinais analógicos de tensão e corrente de maneira sincronizada. A PMU converte estes sinais em dados digitais e fornece valores de magnitude, ângulo de fase, frequência e taxa de variação de frequência – ROCOF (do inglês, *Rate of Change of Frequency*) (IEEE, 2011a; PHADKE; THORP, 2008).

A PMU recebe sincronização e referência de tempo do GPS e envia as informações da rede elétrica com o rótulo de tempo para um concentrador de dados fasoriais – PDC.

O fluxo de informações que descreve o funcionamento de uma PMU é mostrado na Figura 17.

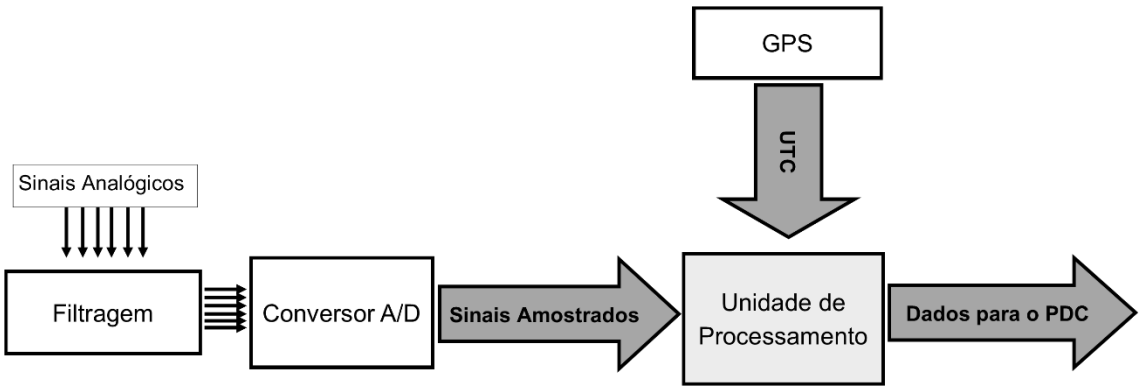

Figura 17. Fluxo de informações de uma PMU (Fonte: Baseado em PHADKE; THORP, 2008).

Os sinais analógicos são filtrados na entrada da PMU para a eliminação de componentes de frequência indesejadas (filtro *anti-aliasing*). Os sinais filtrados passam por um conversor A/D, que gera as amostras a serem processadas pela unidade de processamento da PMU.

A unidade de processamento recebe também as informações de horário UTC e data do receptor GPS.

As amostras são processadas e os fasores são gerados na unidade de processamento. Os dados obtidos são enviados com referência de tempo para o PDC.

A arquitetura de uma PMU pode variar de projeto para projeto. Phadke e Thorp (2008) apresentam uma arquitetura genérica para PMUs, mostrada na Figura 18.

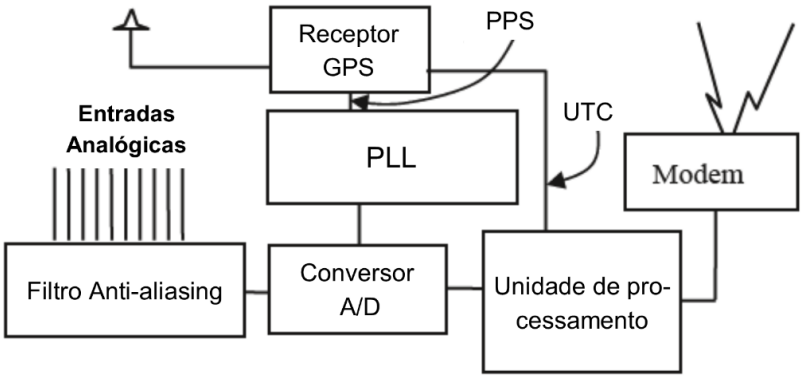

Figura 18. Arquitetura genérica de uma PMU (Fonte: PHADKE; THORP, 2008).

Na arquitetura da Figura 18, o PPS é utilizado como entrada de um circuito PLL (do inglês, *Phase-Locked Loop*). O PLL é um oscilador que é sincronizado com o PPS, e que fornece o pulso de leitura para o conversor A/D na frequência de amostragem utilizada.

Existem outras arquiteturas propostas que apresentam diferenças em relação à arquitetura genérica da Figura 18. É importante ressaltar que esta arquitetura não é um padrão para PMUs definido por norma. A Norma IEEE C37.118.1 não especifica os detalhes de implementação da PMU.

3.6. Concentrador de Dados Fasoriais – PDC

Os concentradores de dados fasoriais – PDCs recebem e armazenam os dados das medições fasoriais de tensões e correntes enviados pelas PMUs, permitindo que os operadores do sistema de energia elétrica tenham acesso às medições sincronizadas de uma grande área geográfica.

Geralmente, nos SPMS, é proposta a arquitetura de comunicação de dados ilustrada na Figura 19, onde existe um PDC regional, que se comunica diretamente com as PMUs de uma dada região; e existe também o Super PDC, que ocupa um nível mais alto na hierarquia da arquitetura de comunicação de dados (CASTELLO *et al*, 2017; PHADKE; THORP, 2008).

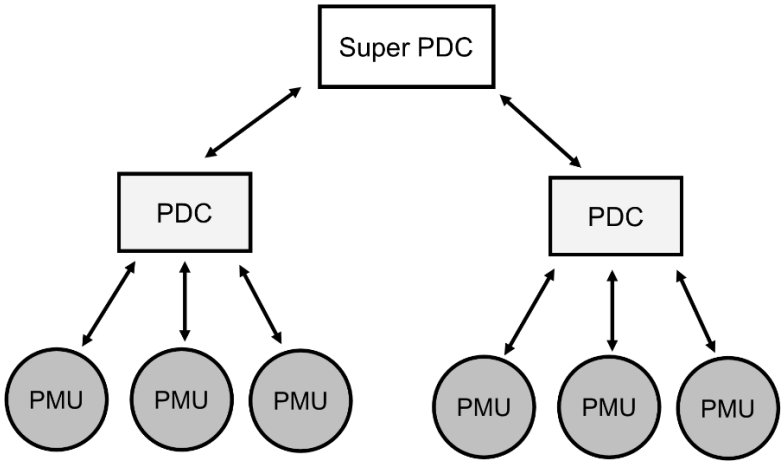

Figura 19. Arquitetura de comunicação de dados dos SPMS (Adaptado de PHADKE, THORP, 2008).

O Super PDC recebe os dados fasoriais de vários PDCs e, desta forma, é capaz de gerar uma grande quantidade de informações sobre o estado do sistema elétrico como um todo. Como os dados obtidos possuem etiqueta de tempo, o operador do sistema pode monitorar a rede em tempo real ou realizar a análise pós-falta do sistema com boa precisão.

3.7. Norma IEEE C37.118 para PMUs

A Norma IEEE C37.118 de 2011 fornece especificações, definições e requisitos de desempenho para as PMUs. Esta norma foi criada para fornecer um padrão para o projeto, funcionamento e comercialização de dispositivos de medição fasorial sincronizada de diferentes fabricantes.

A Norma IEEE C37.118 é dividida em duas partes: A IEEE C37.118.1, cujo título é "*IEEE Standart for Synchrophasor Measurement for Power Systems*" ("Norma IEEE para medição de sincrofasores para os sistemas de potência", em tradução livre), que define as recomendações e procedimentos para a medição de sincrofasores, frequência e ROCOF sob todas as condições de operação do sistema de potência (IEEE, 2011a); e a IEEE C37.118.2, cujo título é "*IEEE Standart for Synchrophasor Data Transfer for Power Systems*" ("Norma IEEE para a transferência de dados de sincrofasores para os sistemas de potência", em tradução livre), que define como deve ser feita a comunicação de dados e trocas de mensagens entre as PMUs e PDCs (IEEE, 2011b).

Uma atualização foi feita na norma IEEE C37.118.1 no ano de 2014, para modificar alguns requisitos de performance da versão original da norma (IEEE, 2014).

3.7.1. Norma IEEE para medição de sincrofasores

A Norma IEEE C37.118.1 de 2011 fornece (IEEE, 2011a):

- Definição dos parâmetros obtidos na medição fasorial sincronizada, tais como: sincrofasores, frequência e ROCOF;
- Métodos para obter e quantificar as medições;
- Definição de testes de performance de PMUs;
- Limites aceitáveis de performance de PMUs.

Esta norma recomenda que a PMU receba a sincronização por tempo de uma fonte confiável e precisa, como o GPS. Além disso, a PMU deve ser capaz de calcular e enviar estimativas de sincrofasores de tensão e/ou corrente de sistemas monofásicos ou trifásicos.

As medições realizadas pela PMU apresentam diferenças em relação aos valores teóricos de magnitude e fase dos fasores. Estas diferenças geram erros de medição, que são quantificados através do TVE (do inglês, "*Total Vector Error"*), definido na Equação 11, que calcula a diferença normalizada entre o fasor teórico e o fasor medido (IEEE, 2011a).

$$\mathrm{TVE(n)} = \sqrt{\frac{\left(\hat{\mathrm{X}}_\mathrm{r}(\mathrm{n})-\mathrm{X}_\mathrm{r}(\mathrm{n})\right)^2+\left(\hat{\mathrm{X}}_\mathrm{i}(\mathrm{n})-\mathrm{X}_\mathrm{i}(\mathrm{n})\right)^2}{\left(\mathrm{X}_\mathrm{r}(\mathrm{n})\right)^2+\left(\mathrm{X}_\mathrm{i}(\mathrm{n})\right)^2}} \qquad \text{(Eq. 11)}$$

Onde $\hat{\mathrm{X}}_\mathrm{r}(\mathrm{n})$ e $\hat{\mathrm{X}}_\mathrm{i}(\mathrm{n})$ são as partes real e imaginária, respectivamente, do n-ésimo fasor estimado em teste pela PMU; $\mathrm{X}_\mathrm{r}(n)$ e $\mathrm{X}_\mathrm{i}(n)$ são a partes real e imaginária do n-ésimo fasor teórico.

As medições de frequência devem ser avaliadas pela Equação 12 (IEEE, 2011a), que define o FE (do inglês, *Frequency Error*). As medições de ROCOF são avaliadas através do RFE (do inglês, *Rate of Change of Frequency Error*), que é definido na Equação 13 (IEEE, 2011a).

$$\mathrm{FE} = |\mathrm{f_{real}} - \mathrm{f_{medido}}| \qquad \text{(Eq. 12)}$$

$$\mathrm{RFE} = |(\mathrm{df/dt})_{\mathrm{real}} - (\mathrm{df/dt})_{\mathrm{medido}}| \qquad \text{(Eq. 13)}$$

A Norma IEEE C37.118 define duas classes de desempenho de PMUs: A **classe M** e a **classe P**. A classe P é utilizada em aplicações que exigem resposta rápida, mas não exigem necessariamente alta precisão.

A classe M é utilizada em aplicações onde existem muitas perturbações da rede e existência de harmônicos. Esta classe fornece alto grau de precisão e apresenta taxas mais baixas de reporte de dados. O protótipo de PMU proposto neste trabalho é de classe P.

Os requisitos de resultados de testes para PMUs classe P para os valores de TVE, FE e RFE são apresentados na Tabela 3 (IEEE, 2014).

Tabela 3. Requisitos para PMUs classe P.

Métrica	**Valor Máximo**
TVE	1%
FE	0,005 Hz
RFE	0,4 Hz/s

Outro ponto abordado na Norma IEEE C37.118.1 é a taxa de reporte exigida para a PMU. A Tabela 4 mostra as taxas de relatório de dados recomendadas para os sistemas de 50 Hz e 60 Hz. Um *frame* corresponde ao conjunto de dados formado por sincrofasor, frequência e ROFOC que correspondem à mesma etiqueta de tempo.

Tabela 4. Taxas de relatório recomendadas pela Norma IEEE C37.118.1 (Adaptado de IEEE, 2011).

Frequência do sistema	**50 Hz**			**60 Hz**					
Taxas de relatório (Fs - *frames* por segundo)	10	25	50	10	12	15	20	30	60

Os *frames* gerados pela PMU devem ocorrer espaçados uniformemente dentro do período de 1 segundo.

3.7.2. Norma IEEE para comunicação de dados

A Norma IEEE C37.118.2 define métodos e definições para a troca de dados entre PMUs e PDCs. Define também os formatos de mensagens entre os dispositivos e os tipos de mensagens trocadas, especificando o conteúdo das mesmas (IEEE, 2011b).

Existem quatro tipos de mensagens definidas na Norma IEEE C37.118.2: dados, configuração, cabeçalho e comando (IEEE, 2011b). Cada tipo é brevemente descrito a seguir:

- Dados: são as medições realizadas pela PMU;

- Configuração: são mensagens que descrevem os tipos de dados, fatores de calibração, entre outros metadados;

- Cabeçalho: são informações descritivas que são lidas e geradas pelo usuário;

- Comando: são códigos que contêm instruções de controle e configuração;

O formato geral das mensagens trocadas (*frame*) entre PMUs e PDCs é ilustrado na Figura 20.

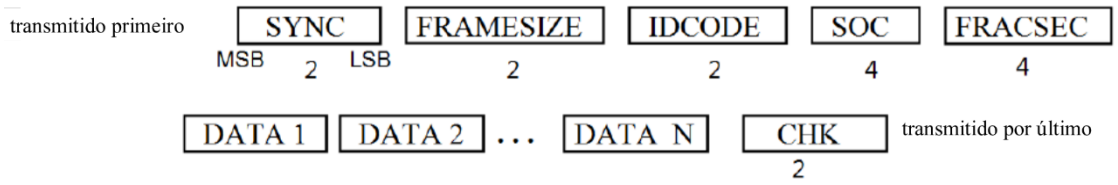

Figura 20. Formato das mensagens trocadas entre PMUs e PDCs (Adaptado de IEEE, 2011b).

O número sob cada campo corresponde ao tamanho do campo em bytes. Por exemplo, o campo IDCODE possui 2 bytes. O campo SYNC é o primeiro a ser transmitido, e o campo CHK é o último a ser transmitido. Cada campo é descrito resumidamente como segue:

- SYNC: é uma palavra de sincronização do *frame*. O primeiro byte é AA em hexadecimal e o segundo byte contém informações de tipo e versão do *frame*;

- FRAMESIZE: número total de bytes contidos no *frame*;

- IDCODE: identifica o destinatário das mensagens para o caso de mensagens de comando, e identifica o remetente para mensagens de outro tipo;

- SOC: etiqueta de tempo fornecida no sistema SOC (do inglês, *Second of Century*), que consiste no número de segundos contados a partir da meia-noite de 01 de janeiro de 1970;

- FRACSEC: fração do segundo em que foram obtidas as medições;

- DATA: medições realizadas pela PMU;

- CHK: número para verificação de erros de transmissão.

O *frame* utilizado no protótipo proposto neste trabalho, utiliza uma versão modificada do *frame* de dados da Figura 20.

3.8. Síntese

Neste capítulo, os principais conceitos sobre os Sistemas de Medição Fasorial Sincronizada foram abordados.

As partes que constituem os SPMS foram detalhadas, e as principais métricas de avaliação de desempenho de PMUs fornecidas pela Norma IEEE C37.118 foram expostas.

4. PROPOSTA DE UMA PMU DE BAIXO CUSTO COM RASPBERRY PI 3

Neste capítulo, a proposta de uma PMU com Raspberry Pi 3 é apresentada. A arquitetura contendo todos os blocos funcionais do dispositivo é descrita, assim como a modelagem de *hardware*, que consiste no circuito para aquisição dos sinais analógicos, e a modelagem de *software*, que consiste no processamento digital dos sinais e envio de informações fasoriais via rede local.

4.1. Arquitetura proposta

A arquitetura proposta neste trabalho para uma PMU de baixo custo com Raspberry Pi 3 é ilustrada na Figura 21.

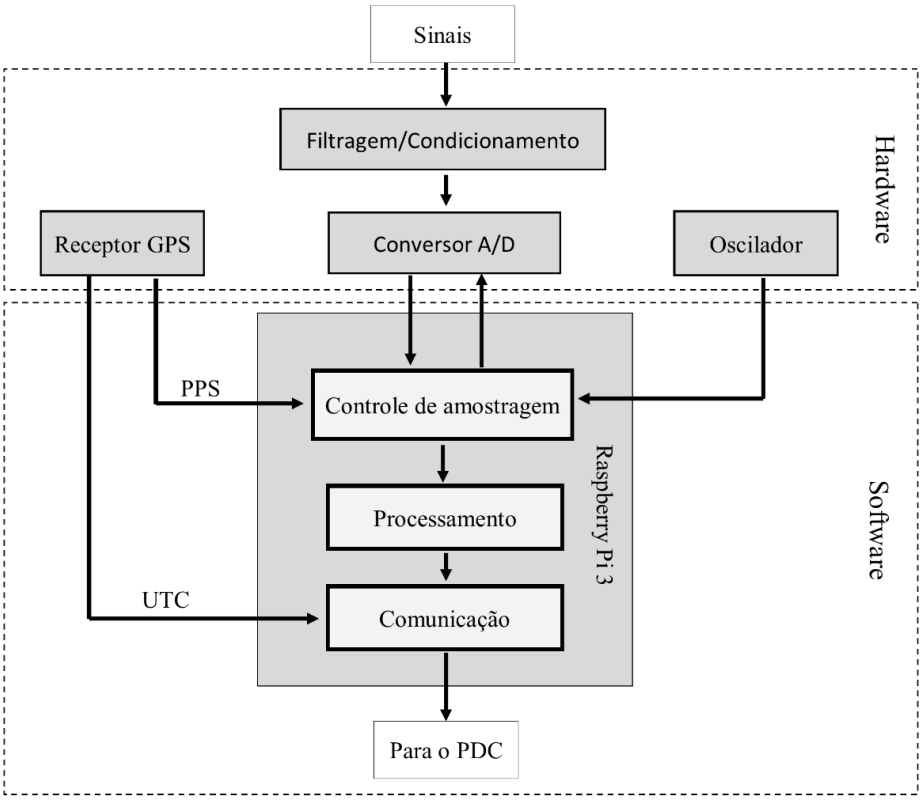

Figura 21. Arquitetura proposta para uma PMU de baixo custo (Fonte: Autor).

Os blocos da arquitetura da Figura 21 são descritos como segue:

a. *Filtragem/condicionamento de sinal*

Este bloco realiza a filtragem do sinal de entrada. Um filtro passa-baixas é implementado com frequência de corte $0{,}5 \cdot f_s$ Hz, onde f_s é a frequência de amostragem utilizada.

O condicionamento dos sinais de entrada da rede elétrica é realizado para que os sinais de entrada do conversor A/D sejam inteiramente positivos e definidos em um intervalo de amplitude de 0,0 V a 3,3 V.

Transformadores de potencial (TPs) são utilizados nos circuitos de condicionamento de sinais de tensão, e transformadores de corrente (TCs) são utilizados nos circuitos de condicionamento de sinais de corrente.

b. *Conversor A/D*

O conversor A/D realiza a amostragem dos sinais de entrada mediante o recebimento do comando de leitura fornecido pelo Raspberry Pi 3. O comando de leitura é enviado pelo Raspberry Pi 3 a partir da detecção da borda de subida do sinal do oscilador. Desta forma, a cada pulso do oscilador uma amostra do conversor A/D é obtida.

c. *Receptor GPS*

O receptor GPS gera a informação de data e horário UTC necessários para etiquetar as medições de tensão e corrente da rede. O receptor GPS também é responsável por enviar o PPS para sincronização das leituras.

d. *Oscilador*

O oscilador gera um pulso na frequência de amostragem pretendida que é utilizado como interrupção externa no Raspberry Pi 3. A cada interrupção, o Raspberry Pi 3 envia um comando para o conversor A/D, que realiza uma amostragem de cada canal.

e. *Controle de amostragem*

O bloco de controle de amostragem representa o envio dos comandos de leitura ao conversor A/D a cada pulso gerado pelo oscilador.

f. *Processamento*

O bloco de processamento representa o processamento das amostras no Raspberry Pi 3. O processamento é realizado através do algoritmo de DFT (Transformada Discreta de Fourier).

Outra função deste bloco é realizar correções das estimativas dos fasores fornecidas pela DFT. Estas correções são necessárias devido aos atrasos de comunicação do conversor A/D.

g. *Comunicação*

O bloco de comunicação é responsável por receber os dados fasoriais da unidade de processamento, estampar estes dados com a etiqueta de tempo fornecida pelo GPS, e enviar a informação gerada para um PDC.

Os blocos de processamento, controle de amostragem e comunicação são implementados via *software* no Raspberry Pi 3, através de programação em *multithreading*.

4.2. Modelagem dos circuitos de aquisição de sinais

Os sinais de tensão e corrente provenientes da rede elétrica necessitam receber um tratamento para que possam ser lidos pelo conversor A/D. Circuitos de aquisição de tensão e corrente são propostos e descritos a seguir.

4.2.1. Aquisição de tensão

O circuito proposto para aquisição dos sinais de tensão da rede elétrica é demonstrado no diagrama de blocos da Figura 22.

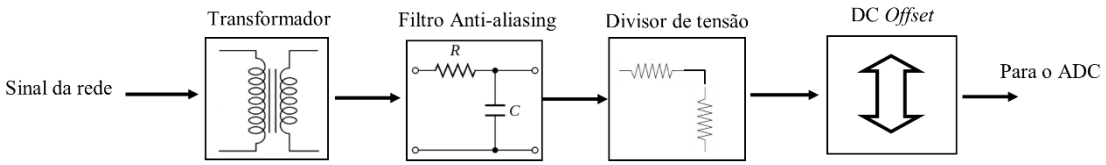

Figura 22. Diagrama de blocos do circuito de aquisição de sinais (Fonte: Autor).

A função de cada bloco no circuito é descrita como segue.

a. *Transformador*

O transformador é utilizado para reduzir os níveis de tensão da rede. Este tipo de transformador não possui a incumbência de fornecer potência ao circuito, desta forma, este transformador é comumente denominado Transformador de Potencial – TP.

b. *Filtro anti-aliasing*

O filtro anti-aliasing consiste em um filtro passivo formado por um resistor em série e um capacitor em paralelo. Este filtro é necessário para prevenir a ocorrência de *aliasing* dos sinais.

c. *Divisor de tensão*

É utilizado para ajustar o nível de tensão de entrada do conversor A/D.

d. *DC Offset*

Este bloco adiciona uma componente DC ao sinal proveniente do filtro anti-aliasing. O objetivo desta adição é realizar um deslocamento (*offset*) vertical positivo no sinal, de forma que o sinal seja inteiramente positivo na saída. Isto é necessário, pois os conversores A/D disponíveis no mercado não suportam tensão negativa em seus terminais.

4.2.2. Aquisição de corrente

A aquisição de corrente proposta é realizada através de transformadores de corrente – TCs. Os TCs empregados realizam a medição de corrente de maneira não invasiva, ou seja, o circuito da rede elétrica não precisa ser aberto para a inserção do medidor de corrente.

O circuito proposto para a aquisição de sinais de corrente é apresentado em forma de diagrama de blocos na Figura 23.

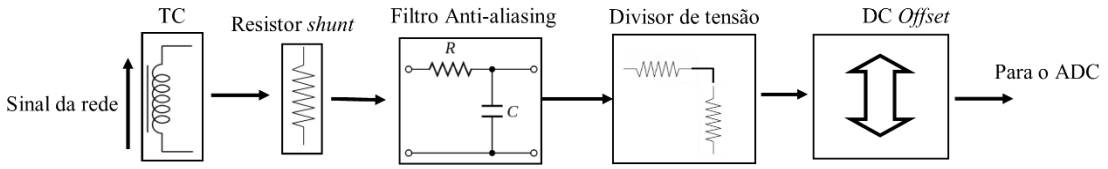

Figura 23. Diagrama de blocos do circuito de aquisição de sinais de corrente (Fonte: Autor).

O diagrama da Figura 23 se diferencia do diagrama de aquisição de tensão (Figura 22) pelos dois primeiros blocos.

O transformador de corrente (TC) é colocado em volta do condutor pelo qual flui a corrente que será medida. O TC retorna uma corrente elétrica em seus terminais que é proporcional à corrente medida no condutor.

O resistor *shunt* é utilizado para transformar a corrente que flui dos terminais do TC em uma tensão elétrica proporcional, através da Lei de Ohm.

4.3. Modelagem do *software*

O modelo de *software* proposto para uma PMU de baixo custo com Raspberry Pi 3 consiste em uma aplicação *multithreading* (várias *threads*) que realiza a amostragem dos sinais, processamento das amostras e envio das informações fasoriais dos sinais de maneira concorrente (em paralelo).

A aplicação proposta contém cinco tarefas (fios de execução) que são realizadas concorrentemente no Raspberry Pi 3: uma interrupção externa para o sinal de PPS, uma interrupção para o oscilador externo que fornece o pulso com o período de amostragem (interrupção de amostragem), uma *thread* de processamento digital dos sinais (*thread* de processamento), uma *thread* para gerar e enviar os *frames* de dados para um servidor em rede local (*thread* de comunicação), e a *main* (*thread* principal), que inicia as comunicações com periféricos e gerencia as demais *threads*.

A Figura 24 mostra os fios de execução que compõem o *software* proposto com as suas respectivas funções. As chamadas de interrupção no Raspberry Pi 3 são tratadas concorrentemente ao programa principal.

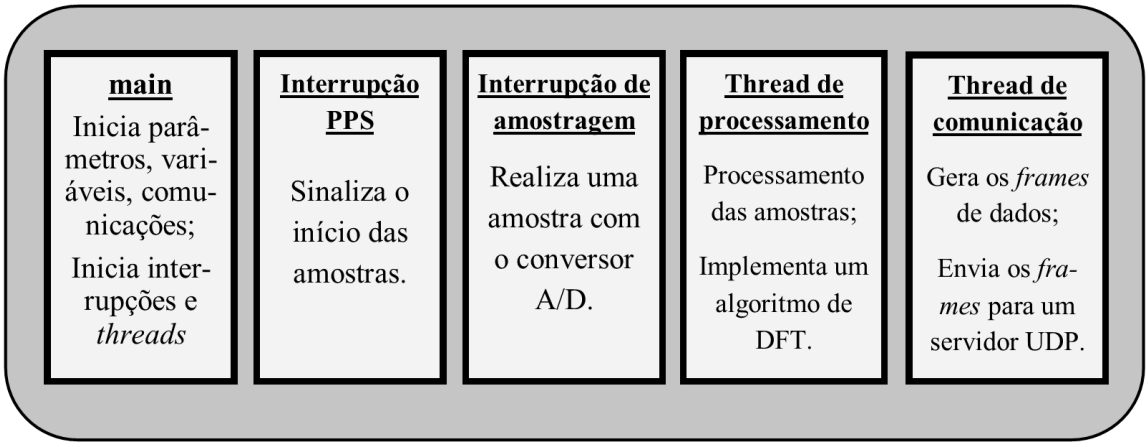

Figura 24. Funções desempenhadas pelas threads que compõem o modelo de software proposto (Fonte: Autor).

A execução da *main* é iniciada antes de todas as *threads* e interrupções, e obtém a data e o horário UTC do receptor GPS. Após desempenhar as suas atribuições, a *main* aguarda até

que a execução das demais tarefas seja terminada. A Figura 25 mostra um diagrama que ilustra o tempo de execução de todas as *threads* da aplicação proposta.

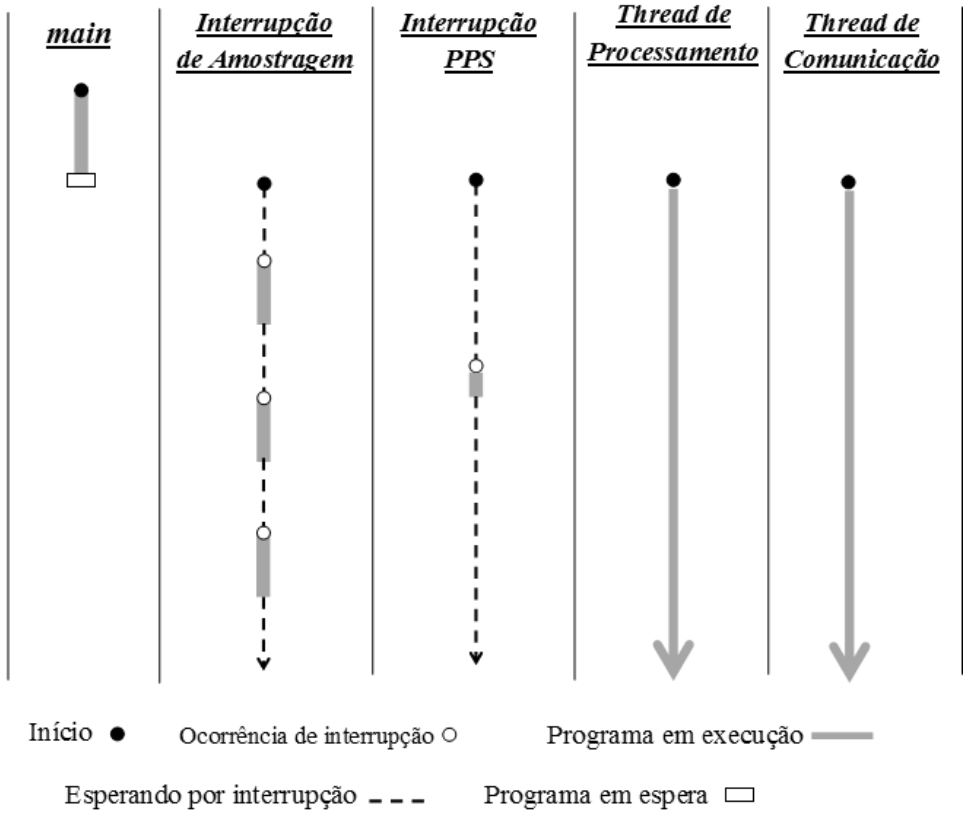

Figura 25. Diagrama de execução das threads que compõem o software proposto (Fonte: Autor).

Observa-se que as *threads* de processamento e comunicação estão sempre em execução. A *thread* de processamento aguarda os dados de leitura do conversor A/D, gerados no tratamento da interrupção de amostragem; e a *thread* de comunicação aguarda os dados gerados na *thread* de processamento. A Figura 26 mostra o fluxo de informações entre as tarefas da aplicação.

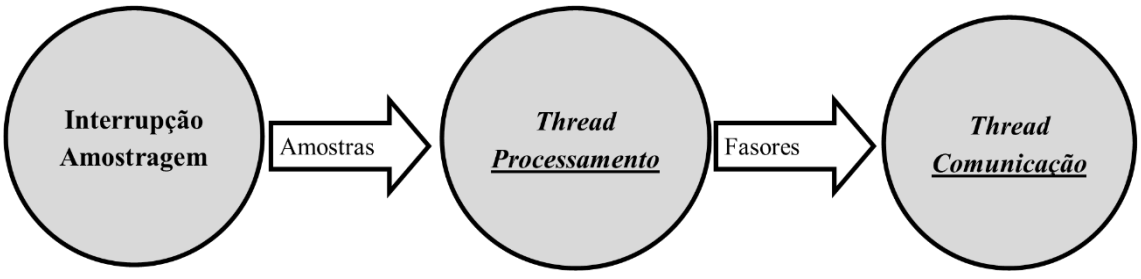

Figura 26. Fluxo de informações da aplicação proposta (Fonte: Autor).

A interrupção do pulso por segundo (PPS) é utilizada para sinalizar ao programa que a próxima amostragem é a primeira amostra do segundo corrente. A primeira amostra após a ocorrência do PPS é marcada, como ilustrado na Figura 27. Desta forma, as leituras das formas de onda de tensão e corrente são sincronizadas com o PPS.

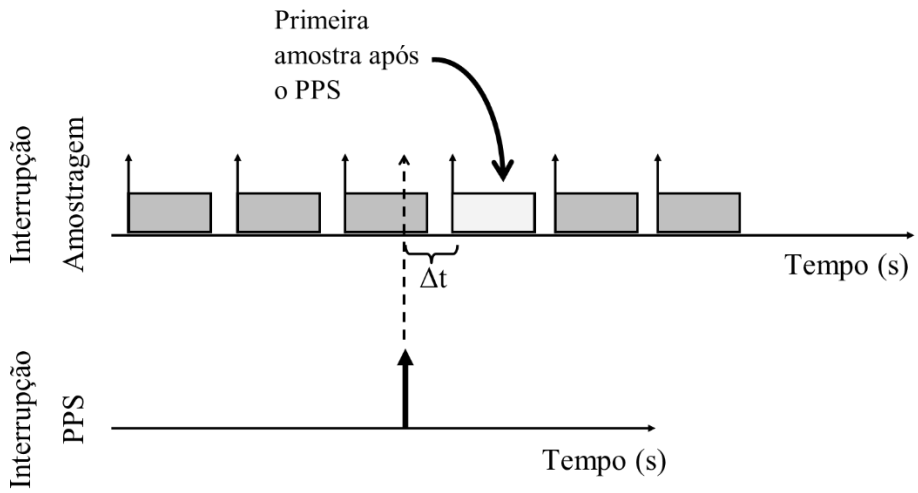

Figura 27. Sinalização das amostras pela interrupção do PPS (Fonte: Autor).

Observa-se na Figura 27 que existe um atraso Δt entre a ocorrência do PPS e a realização da próxima amostragem. Este atraso é de no máximo T_S segundos, onde T_S é o período de amostragem escolhido. É proposta uma compensação deste atraso através de um decremento correspondente a Δt (valor correspondente em radianos) no ângulo de fase dos fasores obtidos.

A técnica utilizada no *software* proposto para sincronização e compartilhamento de dados entre as *threads* segue o modelo produtor-consumidor com *buffer* limitado de dados, abordado na seção 2.3.3 do capítulo 2. O modelo de algoritmo da Figura 8 é utilizado.

É proposto um ***buffer* de amostras**, cujo produtor é a interrupção do oscilador externo (interrupção de amostragem) e o consumidor é a *thread* de processamento; e é proposto um ***buffer* de fasores**, cujo produtor é a *thread* de processamento e o consumidor é a *thread* de comunicação.

O esquema de compartilhamento de dados entre as *threads* da aplicação proposta é ilustrado na Figura 28.

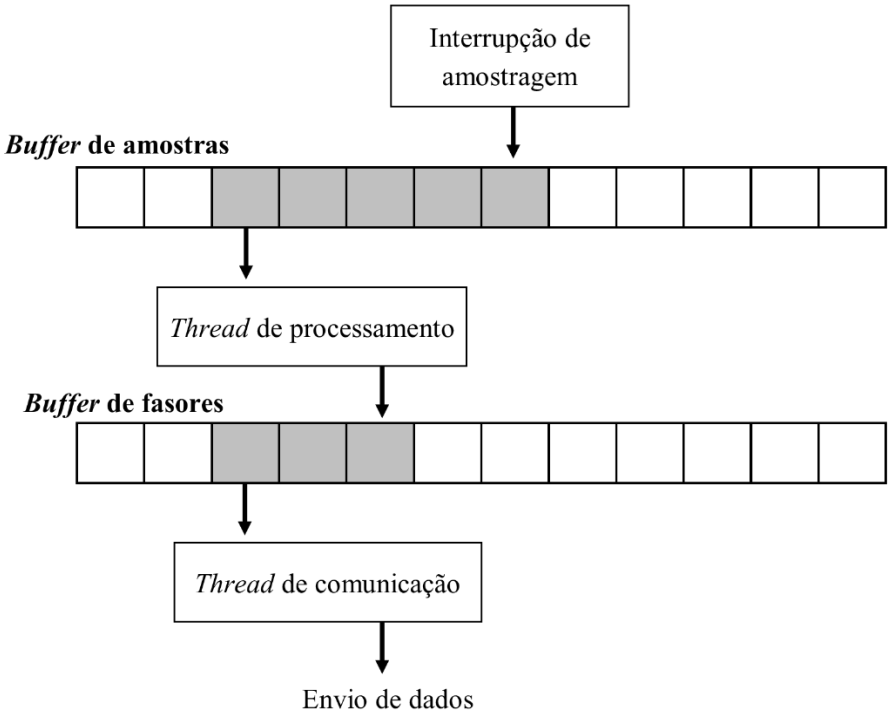

Figura 28. Compartilhamento de dados entre as *threads* da aplicação proposta (Fonte: Baseado em LOPES, 2009).

Na Figura 28, a seta que entra em cada *buffer* indica a produção de dados, enquanto que a seta que sai de cada *buffer* indica o consumo de dados. A *thread* de processamento é consumidora do *buffer* de amostras e produtora do *buffer* de fasores simultaneamente.

O método utilizado de preenchimento e esvaziamento dos *buffers* é o FIFO (do inglês, *First In, First Out*), que define que o primeiro elemento a ser colocado no *buffer* será o primeiro a ser retirado. O produtor de cada *buffer* vai se deslocando para a direita à medida em que vai colocando dados no *buffer*; da mesma forma, o consumidor se desloca para a direita à medida em que vai lendo os dados do *buffer*.

O esquema de utilização de *buffers* ilustrado na Figura 28 permite que a *thread* de processamento execute um algoritmo de DFT sem a necessidade de interromper a amostragem do sinal, já que estas tarefas ocorrem em paralelo.

Como existem dois *buffers*, dois semáforos são criados para sincronizar o acesso das tarefas a cada *buffer*. A sincronização do acesso aos dois *buffers* é apresentada nos fluxogramas da Figura 29.

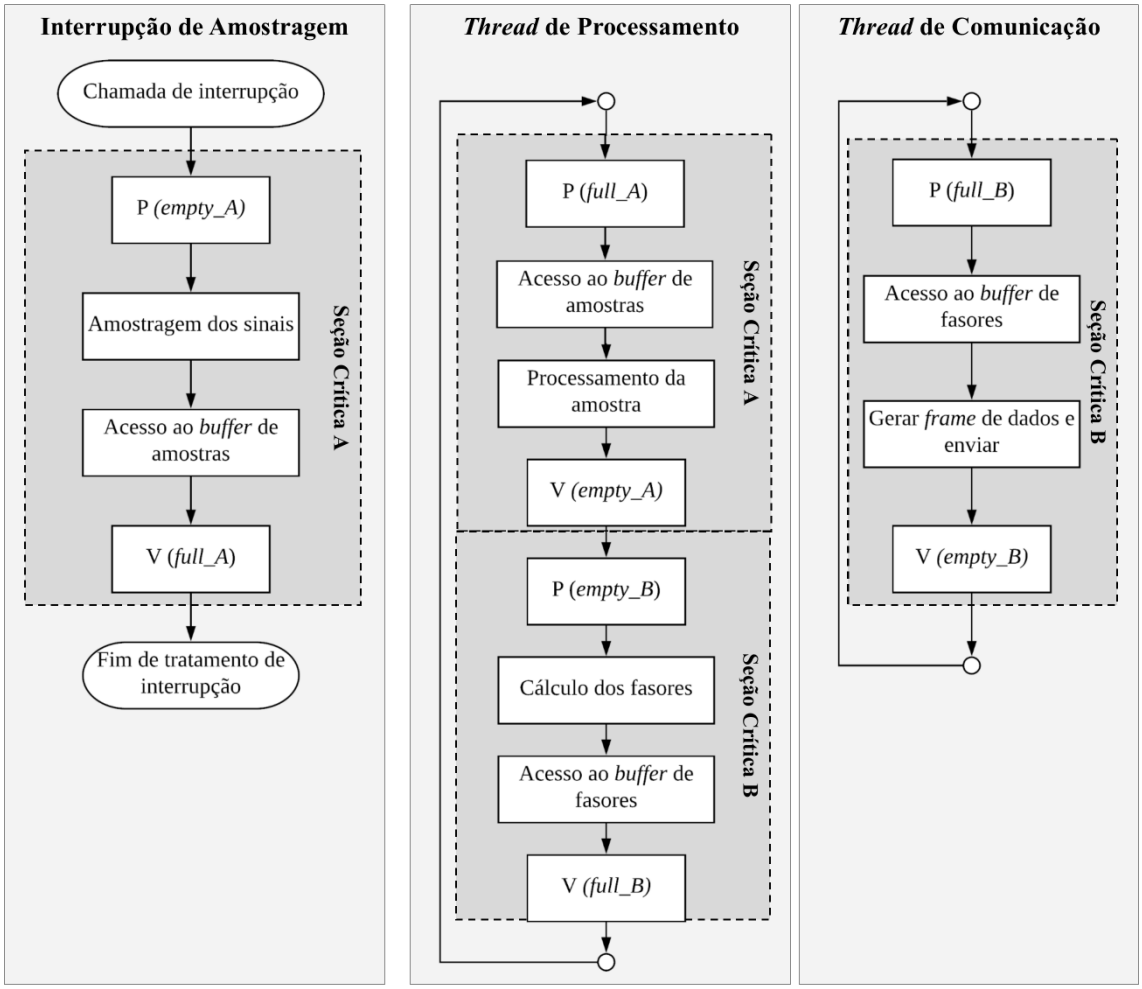

Figura 29. Sincronização para acesso aos *buffers* da aplicação proposta (Fonte: Autor).

A seção crítica A corresponde ao compartilhamento do *buffer* de amostras entre a interrupção de amostragem e a *thread* de processamento; a seção crítica B corresponde ao compartilhamento do *buffer* de fasores entre as *threads* de processamento e comunicação.

4.4. Síntese

A proposta de uma Unidade de Medição Fasorial Sincronizada (PMU) com Raspberry Pi 3 foi apresentada.

A arquitetura proposta é composta de um receptor GPS, um circuito de filtragem e condicionamento de sinais, um conversor A/D e um oscilador configurado para gerar um pulso na frequência de amostragem requisitada. O Raspberry Pi 3 é utilizado para realizar toda a concentração e processamento de dados fornecidos pelo conversor A/D e pelo receptor GPS.

Os circuitos para condicionamento dos sinais de tensão e corrente foram descritos através de diagramas de blocos. Os sinais provenientes da rede elétrica passam por um processo de filtragem e recebem um *offset* (deslocamento vertical) para garantir que não haja sinais negativos nas entradas do conversor A/D.

O modelo de *software* proposto para a PMU com Raspberry Pi 3 foi apresentado. As *threads* e interrupções que compõem o programa foram descritos, assim como os mecanismos utilizados para sincronização e compartilhamento de dados entre as tarefas.

5. PROTOTIPAGEM E TESTES

Neste capítulo, o protótipo desenvolvido de uma PMU de baixo custo com o Raspberry Pi 3 é apresentado. As especificações de funcionamento do protótipo são descritas, e os dispositivos utilizados são mostrados.

Os testes realizados em laboratório são apresentados, assim como os principais resultados obtidos.

5.1. Especificações do protótipo

O protótipo desenvolvido para implementar as funcionalidades de uma PMU com Raspberry Pi 3 é mostrado na Figura 30.

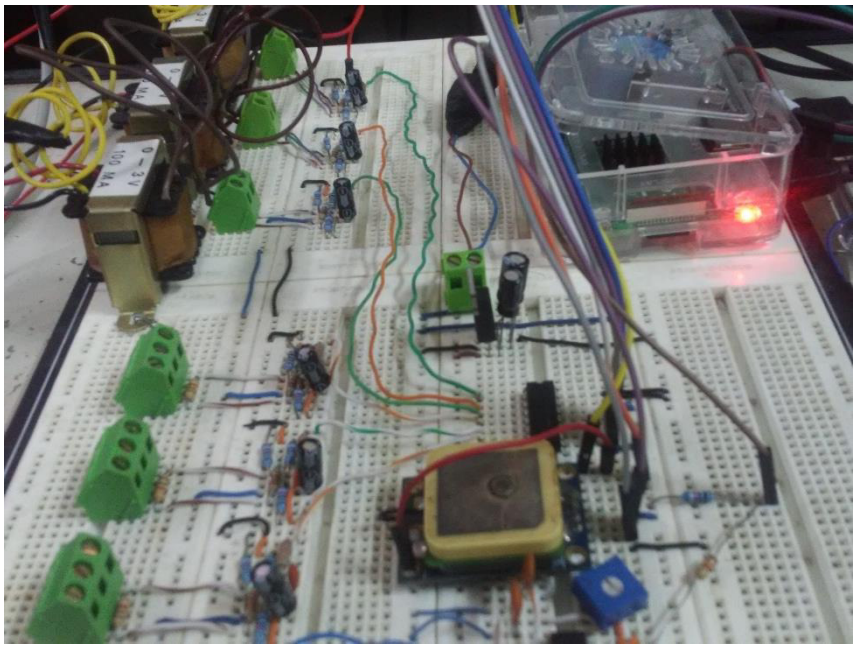

Figura 30. Protótipo de uma PMU de baixo custo com o Raspberry Pi 3.

As principais especificações do protótipo proposto são listadas a seguir:

- Frequência de amostragem (f_S): 1920 Hz;
- Período de amostragem (T_S): 520,83 µs;
- Amostras por ciclo: 32;
- Janela de amostragem composta de 5 ciclos de onda de 60 Hz;

- Número de canais: 3 canais de corrente e 3 canais de tensão;
- Realiza a medição do fasor (valor RMS e fase) dos 6 canais;
- Possui referência de tempo do GPS e insere uma etiqueta de tempo às leituras realizadas;
- As leituras são sincronizadas com o pulso por segundo fornecido pelo módulo GPS;
- Gera um *frame* de dados contendo os 6 fasores medidos e a etiqueta de tempo;
- Envio de até 10 *frames* por segundo (10 fps) a um servidor em rede local via UDP;
- Faixa de tensão de entrada da rede elétrica: 5 V – 230 V RMS;
- Faixa de corrente elétrica de entrada: 0 – 5 A RMS;
- Alimentação do Raspberry Pi 3: 5 V DC @ 2,5 A;
- Alimentação dos circuitos montados na placa e demais dispositivos utilizados: 5 V DC @ 300 mA.

A frequência de amostragem foi escolhida com base no número desejado de amostras por ciclo. A escolha de 32 amostras por ciclo e 60 ciclos por segundo resultam em 1920 amostras por segundo (1920 Hz).

A janela de amostragem selecionada possui 5 ciclos de 60 Hz, que resultam em N = 160 amostras. São necessários 83,33 ms para a aquisição das 160 amostras. Estas 160 amostras são utilizadas para gerar um *frame* contendo os fasores de tensão e corrente dos sinais de entrada da PMU.

Como o protótipo gera 10 *frames* por segundo, o processo de amostragem é realizado 10 vezes seguidas, resultando em uma janela de amostragem total de 833,3 ms a cada segundo. A Figura 31 ilustra a janela de amostragem total do protótipo desenvolvido.

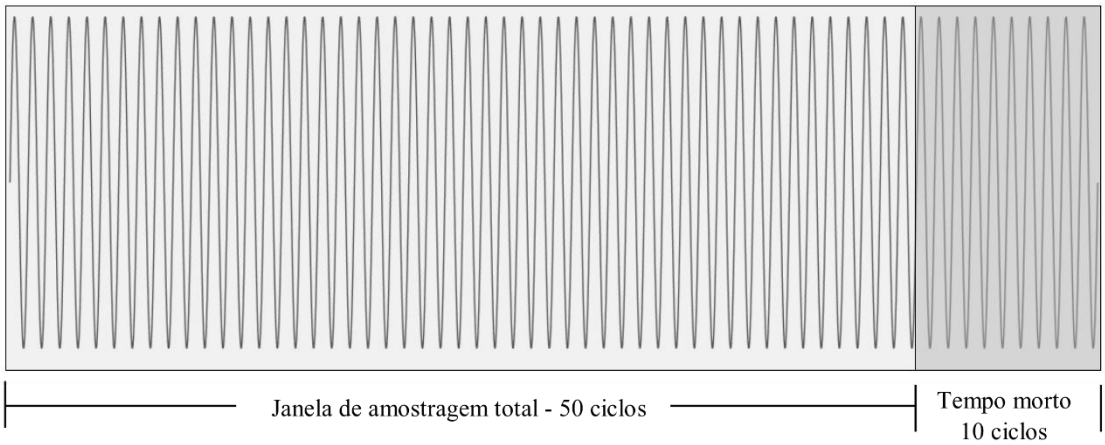

Figura 31. Janela de amostragem do protótipo de uma PMU com Raspberry Pi 3.

A janela de amostragem da Figura 31 corresponde ao período de um segundo. Observa-se que existe um tempo morto de 10 ciclos de 60 Hz (166,67 ms), em que não há aquisição dos sinais de entrada da PMU. Este tempo morto é necessário principalmente para o esvaziamento dos *buffers* de amostras armazenados na memória do Raspberry Pi 3.

5.2. Hardware

O *hardware* proposto do protótipo da PMU consiste em um circuito elétrico montado em uma *protoboard* que realiza as funções de filtragem/condicionamento dos sinais de entrada e ligação elétrica dos dispositivos que compõem a PMU.

5.2.1. Dispositivos empregados

Os dispositivos que compõem o *hardware* do protótipo são listados a seguir.

- Conversor A/D: MCP3008;
- Módulo GPS: GY-GPS6MV2 com Ublox NEO-6M-01-001;
- Oscilador: NE555;
- Regulador de tensão: LM2937 – 3.3V;
- TP: Transformador Bivolt 110/220 V – 3.3 V @ 300 mA;
- TC: Transformador de corrente SCT-013-000.

Conversor A/D MCP3008

Trata-se de um conversor A/D de 8 canais com 10 bits de resolução e interface de comunicação SPI. A Figura 32 mostra o MCP3008 utilizado no protótipo proposto neste trabalho.

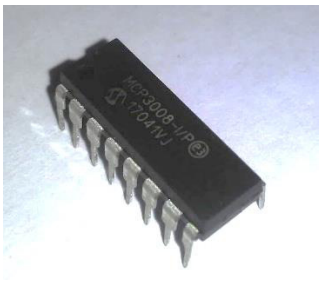

Figura 32. Conversor A/D MCP3008 (Fonte: Autor).

A principais características do MCP3008 são listadas a seguir (MICROCHIP, 2007):

- Resolução de 10 bits;
- Interface serial SPI;
- Alimentação: 2,7 V – 5,5 V;
- Taxa de amostragem máxima de 200 ksps a 5 V;
- Taxa de amostragem máxima de 75 ksps a 2,7 V;
- Corrente típica em *standby* de 5 nA. Máxima de 2 µA;
- Corrente máxima em operação de 500 µA com alimentação em 5 V;
- Faixa de temperatura industrial: -40 º C a +85 º C.

O MCP3008 é encapsulado em um CI (Circuito Integrado) do tipo DIP de 16 pinos. A pinagem do dispositivo e demais informações podem ser encontradas no datasheet disponibilizado pelo fabricante (MICROCHIP, 2007).

A Tabela 5 especifica as conexões com os pinos do MCP3008 utilizadas no protótipo proposto.

Tabela 5. Conexões com os pinos do MCP3008.

Pino MCP3008	Conexão	Pino MCP3008	Conexão
1 (CH0)	Tensão 1	9 (DGND)	Ground
2 (CH1)	Tensão 2	10 (CS)	RPi3 pin 24
3 (CH2)	Tensão 3	11 (Din)	RPi3 pin 19
4 (CH3)	Corrente 1	12 (Dout)	RPi3 pin 21
5 (CH4)	Corrente 2	13 (CLK)	RPi3 pin 23
6 (CH5)	Corrente 3	14 (AGND)	Ground
7 (CH6)	---	15 (Vref)	3,3 V
8 (CH7)	---	16 (Vdd)	3,3 V

Módulo GPS GY-GPS6MV2

O módulo GPS GY-GPS6MV2, mostrado na Figura 33, fornece uma interface de comunicação serial UART ao receptor GPS Ublox NEO-6M-01-001.

Figura 33. Módulo GPS GY-GPS6MV2 (Fonte: Autor).

As principais características do Módulo GPS GY-GPS6MV2 são listadas a seguir:

- Alimentação: 3,3 V a 5,0 V;
- LED indicador de sinal;
- Comunicação serial UART com *baud rate* padrão de 9600 bps;
- Memória interna EEPROM para armazenar as configurações do dispositivo.

Por padrão, este módulo GPS recebe as informações de GPS no padrão NMEA 0186 e envia estas informações através da sua interface UART. Neste protótipo, o Raspberry Pi 3 é configurado para receber as informações do GPS via protocolo UART.

A Tabela 6 especifica as conexões dos pinos do GY-GPS6MV2 utilizadas no protótipo proposto.

Tabela 6. Conexões com os pinos do módulo GPS GY-GPS6MV2.

Pino GY-GPS6MV2	Conexão
Vcc (Alimentação)	3,3 V
GND (Ground)	Ground
RX	RPi3 pin 14
TX	RPi3 pin 15

Oscilador NE555

O NE555 é um oscilador de alta precisão, utilizado para várias aplicações na eletrônica. A Figura 34 mostra o CI NE555.

Figura 34. Oscilador NE555 (Fonte: Autor).

As principais características deste dispositivo são listadas a seguir (ST MICROELETRONICS, 2012):

- Tensão de alimentação: 5,0 V a 15,0 V;
- Corrente máxima de alimentação: 15 mA;
- Operação nos modos monoestável e astável;
- Frequência de oscilação configurável via combinação de capacitor e resistores;
- Faixa de frequência do oscilador no modo astável: 1 mHz a 100 kHz.

A operação em modo astável, utilizada no protótipo proposto, requer a montagem do circuito da Figura 35.

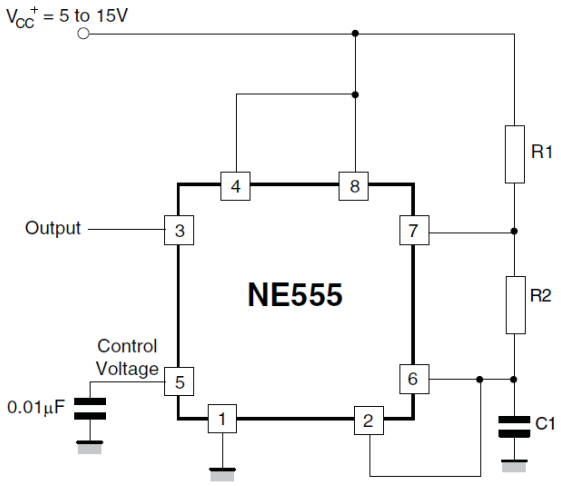

Figura 35. Circuito para a operação em modo astável do 555 (Fonte: ST MICROELETRONICS, 2018).

O capacitor C_1 e os resistores R_A e R_B devem ser determinados para configurar a oscilação do 555 na frequência desejada. As equações 14, 15 e 16 (ST MICROELETRONICS, 2018) são utilizadas para determinar os valores desses elementos a partir do período de oscilação desejado.

$$t_H = 0{,}693(R_1 + R_2)C_1 \quad \text{(Eq. 14)}$$

$$t_L = 0{,}693(R_1)C_1 \quad \text{(Eq. 15)}$$

$$\text{período} = t_H + t_L = 0{,}693(R_1 + 2R_2)C_1 \quad \text{(Eq. 16)}$$

Onde t_L é o tempo do sinal em nível baixo, e t_H é o tempo em nível alto. O período de oscilação desejado é de 520,87 µs. Selecionando um valor de capacitância de 0,1 µF e um valor de resistência R_1 = 1,0 kΩ, obtém-se R_1 = 3,26 kΩ.

Transformador de potencial empregado

O TP empregado é mostrado na Figura 36. Este dispositivo possui uma entrada em 220 V RMS e uma entrada em 110 V RMS. A saída é de 3,3 V RMS com corrente máxima de 300 mA.

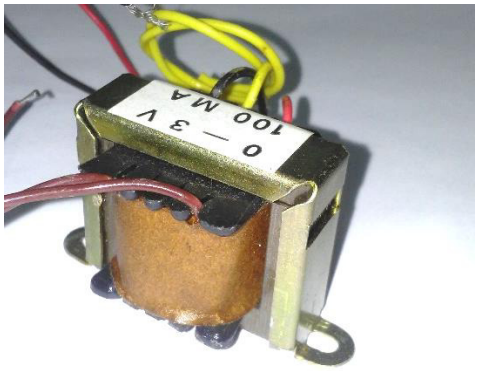

Figura 36. Transformador de potencial empregado no protótipo (Fonte: Autor).

Para o protótipo proposto, o enrolamento de 220 V é utilizado.

Transformador de corrente SCT-013-000

O TC SCT-013-000 é mostrado na Figura 37. Este transformador pode medir correntes elétricas AC de até 100 A RMS de maneira não invasiva.

Figura 37. Transformador de corrente não invasivo SCT-013-000 (Fonte: Autor).

Este dispositivo retorna uma corrente elétrica de até 50 mA (valor RMS) proporcional à corrente medida (até 100 A). Como a obtenção de correntes elétricas próximas de 100 A é impraticável no laboratório utilizado, limitou-se as correntes de carga a 5 A.

5.2.2. Circuito de filtragem e condicionamento dos sinais

O circuito utilizado para filtragem e condicionamento dos sinais de entrada de tensão é ilustrado na Figura 38.

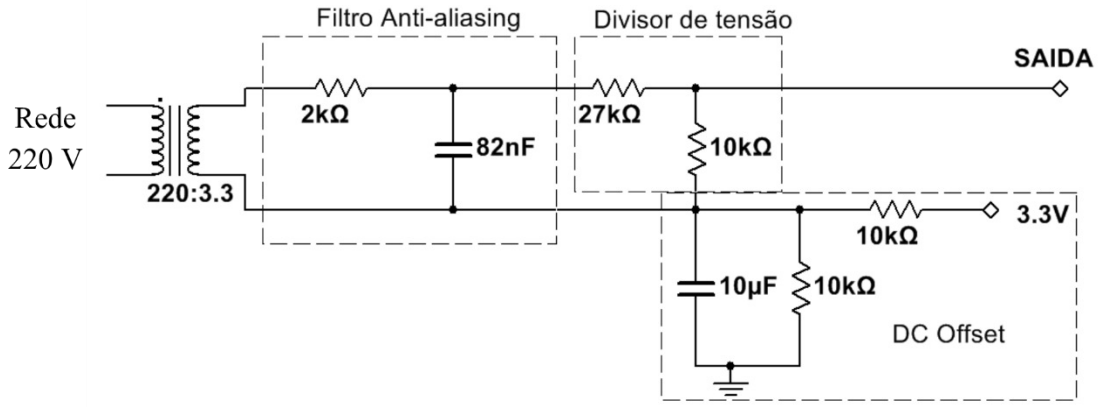

Figura 38. Circuito de filtragem e condicionamento de sinais de tensão.

O circuito de condicionamento de sinais da Figura 38 tem como objetivo manter o valor de tensão abaixo de 3,3 V na entrada do conversor A/D. O *offset* inserido no sinal de entrada é de 1,65 V. A Figura 39 ilustra o condicionamento do sinal da rede.

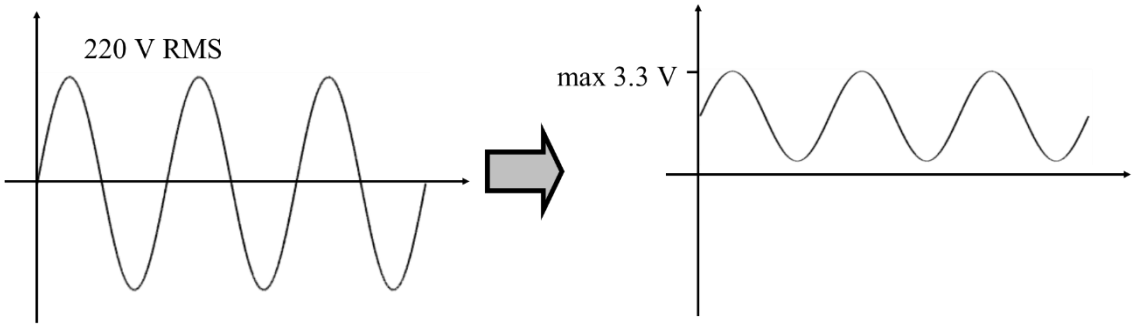

Figura 39. Condicionamento do sinal da rede (Fonte: Autor).

O circuito para aquisição de corrente é idêntico ao circuito da Figura 38. A diferença é que um TC (transformador de corrente) é utilizado com um resistor em paralelo, para converter o sinal de corrente em sinal de tensão. A Figura 40 mostra o circuito utilizado para aquisição de corrente da rede elétrica.

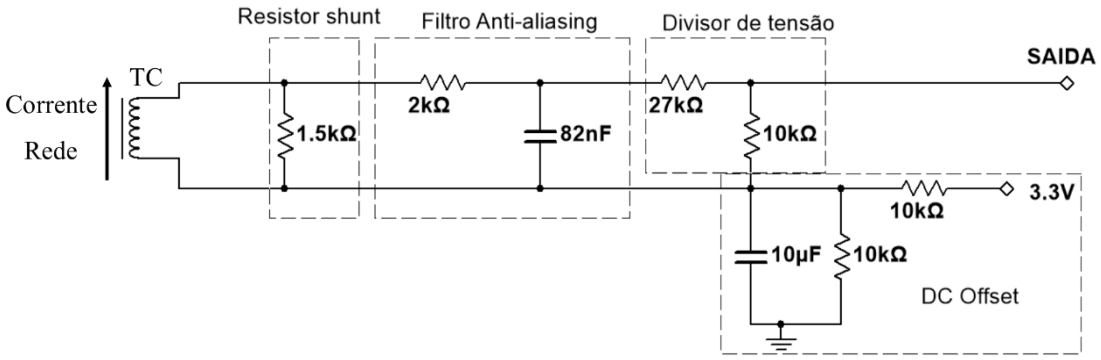

Figura 40. Circuito para aquisição de sinais de corrente.

5.2.3. Montagem do hardware do protótipo na protoboard

O circuito para conexão entre os dispositivos da PMU proposta e para aquisição de sinais da rede elétrica é montado em uma *protoboard*. A Figura 41 mostra o esquema de circuito utilizado para a montagem.

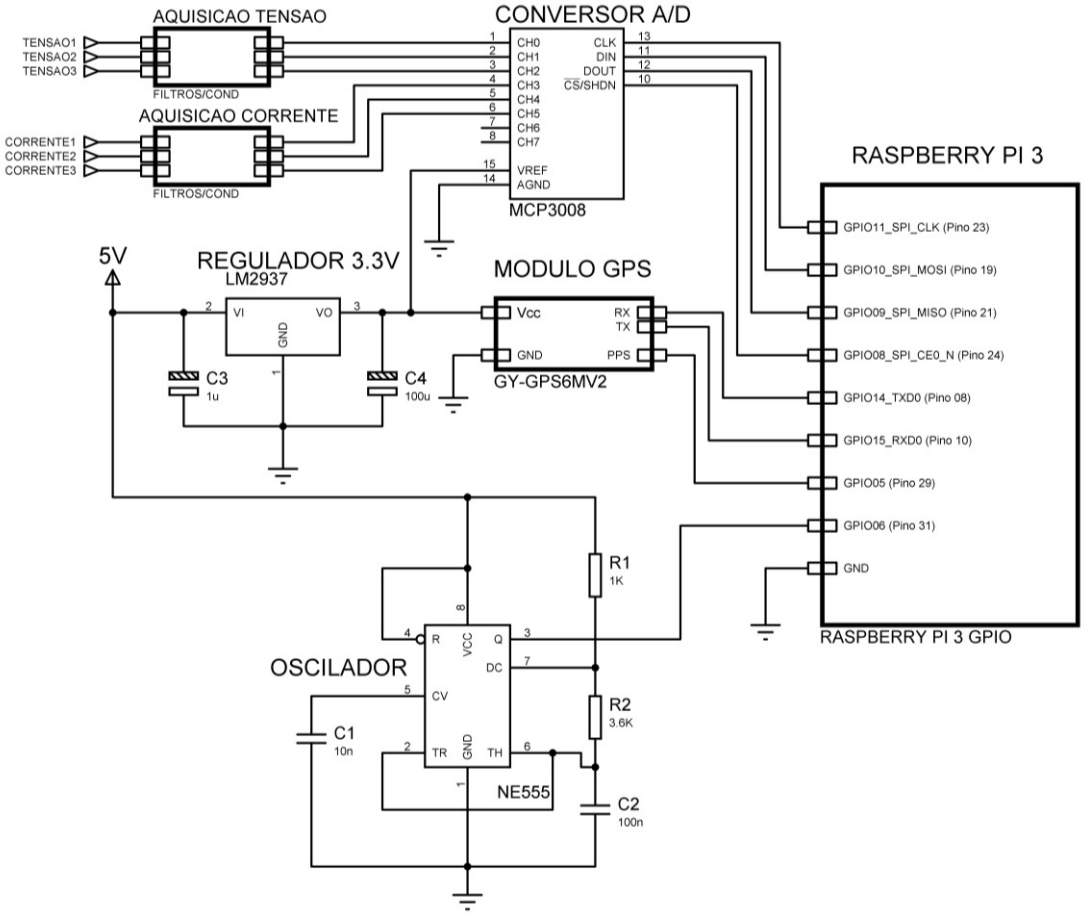

Figura 41. Esquema de circuito do protótipo da PMU.

O circuito montado é alimentado com uma fonte de 5 V DC. Esta fonte deve ser capaz de fornecer pelo menos 100 mA ao circuito. O Raspberry Pi 3 é alimentado com uma outra fonte de 5 V DC, sendo que esta fonte deve ser capaz de fornecer no mínimo 2,0 A.

O regulador de tensão LM2937 é utilizado para regular a tensão de 5 V para 3,3 V. O barramento de 3,3 V é utilizado para alimentar os circuitos de aquisição dos sinais, o módulo GPS e o MCP 3008. O barramento de 5 V é utilizado somente para alimentar o oscilador NE555.

A Figura 42 mostra a montagem do circuito do protótipo da PMU em uma *protoboard*.

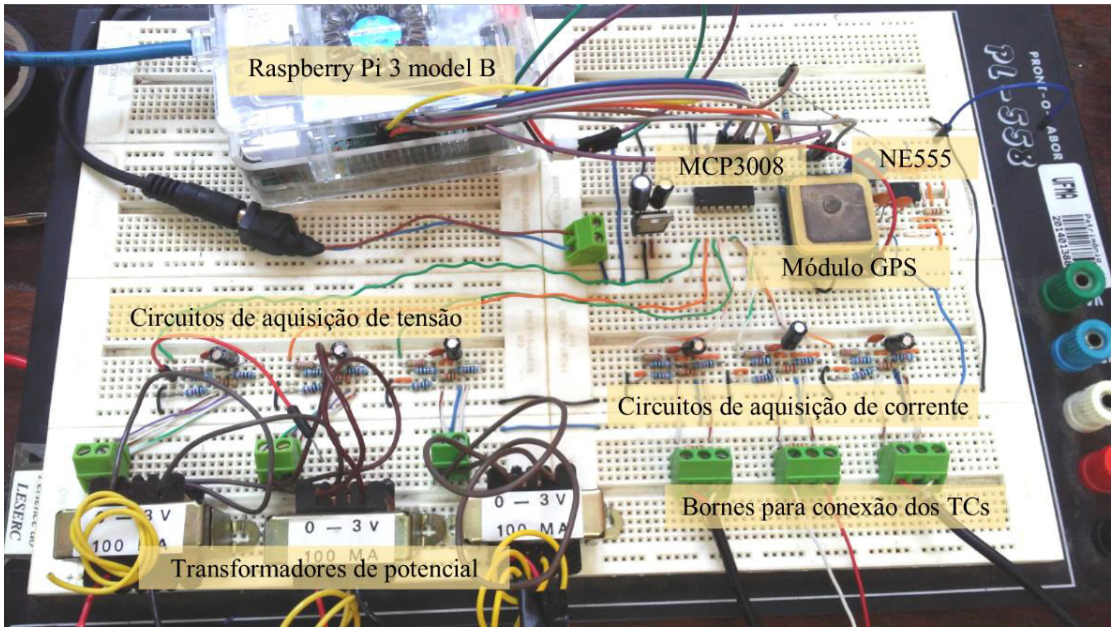

Figura 42. Montagem do protótipo da PMU em protoboard

5.2.4. Custo do protótipo

A Tabela 7 demonstra o custo de cada componente que constitui o protótipo proposto e apresenta também o custo total.

Tabela 7. Demonstrativo do custo do protótipo de uma PMU com Raspberry Pi 3.

Material	Qtde	Custo Unit.	Custo Total
Raspberry Pi 3 Model B	1	R$ 200,00	R$ 200,00
Módulo GPS GY-GPS6MV2	1	R$ 47,50	R$ 47,50
CI NE555	1	R$ 0,90	R$ 0,90
TC SCT 013-000	3	R$ 50,00	R$ 150,00
TP 3V	3	R$ 15,00	R$ 45,00
MCP3008	1	R$ 19,40	R$ 19,40
Resistores de 2K	20	-	R$ 1,90
Resistores de 10K	20	-	R$ 1,90
Resistores de 27K	20	-	R$ 1,90
Resistores de 1,5K	20	-	R$ 1,90
Capacitores eletrolíticos de 10 uF	10	-	R$ 2,40
Capacitores cerâmicos de 82 nF	10	-	R$ 2,00
Capacitores Eletrolíticos de 100 uF	5	-	R$ 2,90
Capacitores Eletrolíticos de 1 uF	5	-	R$ 1,20
Fonte de 5V 600 mA	1	R$ 15,00	R$ 15,00
Bornes e conectores	-	-	R$ 10,00
Protoboard 2420 pontos	1	149	R$ 149,00
Regulador LM2937 (Lote 5 Items)	1	30,73	R$ 30,73
Total	-	-	**R$ 683,63**

5.3. *Software*

O *software* que implementa o modelo apresentado no Capítulo 4 foi desenvolvido em linguagem C. A API de C utilizada para implementar os recursos de *multithreading* é a POSIX Threads. Esta API é incluída no código através do comando #include <pthread.h>. As funcionalidades da biblioteca *pthread.h* são utilizadas para criar e inicializar as *threads* de processamento de amostras e comunicação.

As interrupções de amostragem e do pulso por segundo são configuradas no código em C através da biblioteca *wiringPi*. A função *wiringPiISR(int pin, int mode, void* function)* inicia uma interrupção externa no Raspberry Pi 3. Onde *pin* é o pino utilizado para a interrupção externa, seguindo a numeração do *wiringPi*; *mode* é o tipo de interrupção (borda de subida, borda de descida ou as duas); e *function* corresponde ao endereço da função que será chamada quando ocorrer a interrupção.

A sincronização do acesso das *threads* e interrupções aos recursos críticos (*buffers* limitados) é feita através da API de C *semaphore.h*. Esta biblioteca permite ao programador criar e configurar os semáforos.

Os códigos utilizados para comunicação SPI com o MCP3008, comunicação UART com o GPS Ublox Neo 6M-01-001 e comunicação via UDP com o servidor em rede local (PDC), estão disponíveis no Apêndice A.

5.4. Concentrador de Dados Fasoriais - PDC

O PDC utilizado para os testes deste protótipo consiste em um computador que está conectado à mesma rede LAN que o Raspberry Pi 3. É proposta uma arquitetura cliente-servidor simples, como ilustra a Figura 43, para a troca de informações entre a PMU proposta e o PDC.

Figura 43. Arquitetura cliente-servidor simples para comunicação entre o PMU e o PDC.

O cliente corresponde à *thread* de comunicação no Raspberry Pi 3 que envia os dados ao servidor via UDP. O servidor consiste em um programa escrito em linguagem Python em um computador em rede local que recebe e imprime os dados enviados pela PMU.

O *frame* de dados proposto para o envio de informações da PMU ao PDC é apresentado na Figura 44.

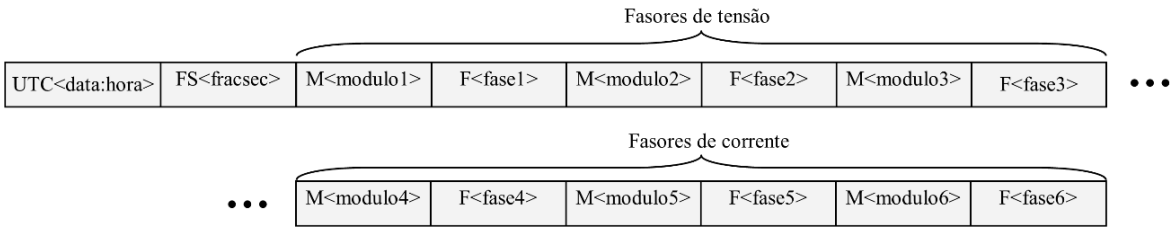

Figura 44. Formato do frame gerado pela PMU (Fonte: Baseado em IEEE, 2011b).

O *frame* de dados da Figura 44 consiste em um conjunto de caracteres ASCII, com comprimento máximo de 64 *bytes*.

5.5. Cenário de testes

Os testes do protótipo foram realizados no Laboratório de Medidas Elétricas. Uma fonte de tensão trifásica a quatro fios (três fases e um neutro) foi conectada a uma carga trifásica equilibrada, como mostra a Figura 45. As correntes de linha e as tensões de fase são medidas.

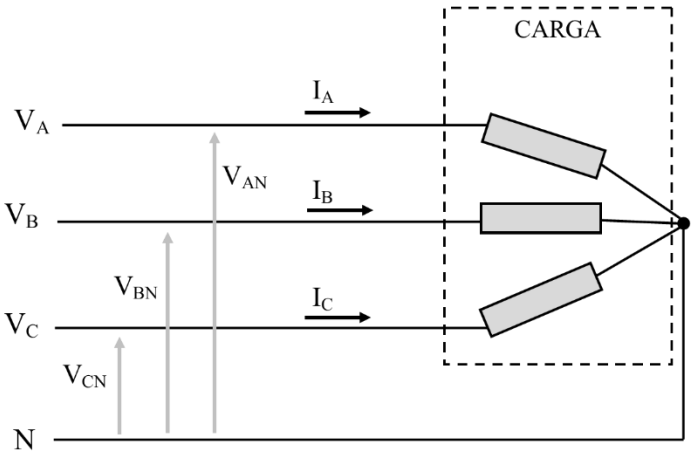

Figura 45. Esquema do circuito montado em laboratório para testes.

A carga utilizada é um conjunto de lâmpadas incandescentes que possui 500 W por fase, 1500 W no total, e é uma carga resistiva. Os valores de referência para as medições são obtidos através de medição de tensão com o multímetro e medição de corrente com o amperímetro. A Figura 46 mostra a montagem em laboratório para a realização dos testes.

Figura 46. Montagem em laboratório para realização de testes.

As medições são realizadas para diversos valores de tensão da fonte trifásica: 200 V, 210 V, 220 V, 230 V, entre outros. A fonte trifásica utilizada no experimento trata-se um varivolt trifásico, mostrado na Figura 47. Para cada valor de tensão, os valores de referência V_{AN}, V_{BN}, V_{CN}, I_A, I_B e I_C são obtidos e em seguida as medições do protótipo da PMU são realizadas. A tensão de entrada é mantida fixa durante o processo de medição.

Figura 47. Varivolt trifásico utilizado nos experimentos.

A medição é realizada pela PMU durante um período de alguns minutos para cada valor de tensão de entrada. Os dados colhidos são analisados com o auxílio do MATLAB. Os valores de magnitude e fase são plotados em função do tempo e as formas de onda da rede são reconstituídas através das informações fasoriais obtidas. Os resultados são comparados com os valores de referência.

5.6. Resultados dos testes

Os resultados são apresentados em forma de tabelas, contendo os valores de tensão e corrente lidos, juntamente com a etiqueta de tempo fornecida pelo GPS. Os gráficos das formas de onda de tensão e corrente são gerados no MATLAB, assim como os diagramas fasoriais.

5.6.1. Tensão de entrada em 220 V

A fonte trifásica foi ajustada em 220 V RMS. Os valores de referência obtidos com multímetro e amperímetro estão na Tabela 8.

Tabela 8. Valores de referência para o teste com tensão da fonte em 190 V.

Grandeza	Valor (V)	Grandeza	Valor (A)
Va	221,6	Ia	2,1
Vb	223,3	Ib	2,1
Vc	223,5	Ic	2,1

Os dados medidos pela PMU são recebidos no PDC em formato de texto. Estes dados podem ser acessados através de um editor de texto simples, como mostra a Figura 48.

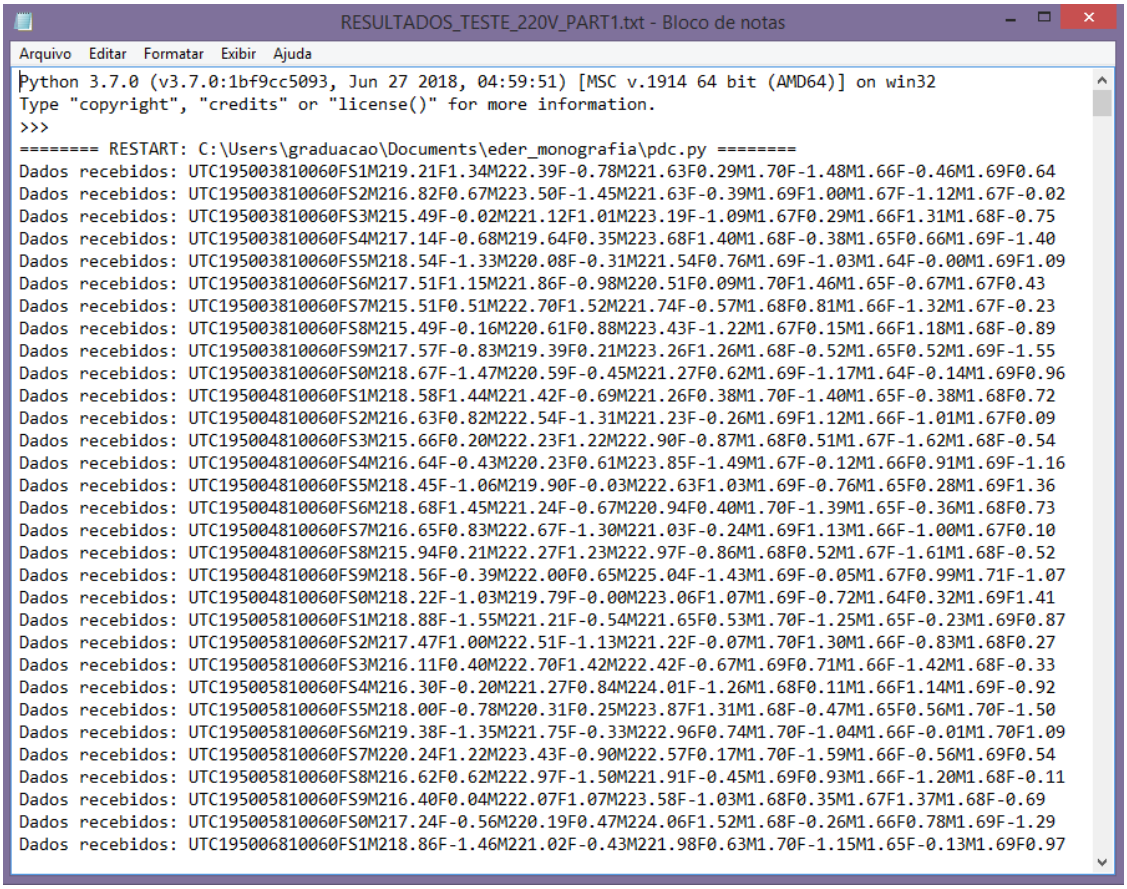

Figura 48. Dados da PMU recebidos no PDC em formato de texto.

A Tabela 9 mostra, de maneira organizada, as medições realizadas pela PMU durante o período de dois segundos.

Tabela 9. Medições realizadas pelo protótipo de PMU.

Horário/Data UTC	FS	Mag Va	Fase Va	Mag Vb	Fase Vb	Mag Vc	Fase Vc	Mag Ia	Fase Ia	Mag Ib	Fase Ib	Mag Ic	Fase Ic
UTC195341280618	1	219,76	-177,71	224,75	61,31	226,35	-58,44	1,7	-160,52	1,68	78,50	1,7	-39,53
UTC195341280618	2	219,57	158,23	223,67	37,82	226,38	-81,93	1,69	175,99	1,67	55,00	1,71	-63,03
UTC195341280618	3	220,55	134,16	223,57	13,18	226,34	-106,09	1,7	151,35	1,67	30,94	1,71	-87,09
UTC195341280618	4	220,68	110,67	223,66	-10,31	225,56	-129,58	1,7	127,86	1,67	7,45	1,71	-110,10
UTC195341280618	5	221,28	87,66	224,89	-33,23	225,6	-152,50	1,71	105,52	1,68	-15,47	1,71	-133,02
UTC195341280618	6	222,53	65,89	226,41	-55,00	226,71	-173,70	1,72	84,80	1,68	-36,10	1,71	-152,50
UTC195341280618	7	219,92	42,97	225,29	-78,50	225,4	162,24	1,71	60,16	1,68	-61,31	1,7	-178,85
UTC195341280618	8	219,79	21,77	225,01	-99,21	225,89	141,04	1,7	38,96	1,68	-82,51	1,7	159,95
UTC195341280618	9	219,62	0,57	224,48	-120,41	226,23	119,84	1,7	17,76	1,68	-103,22	1,7	138,75
UTC195341280618	0	220,02	-21,20	224,16	-142,18	226,3	98,07	1,7	-4,01	1,68	-124,42	1,71	116,97
UTC195342280618	1	220	-173,70	224,91	65,32	226,24	-54,43	1,7	-155,94	1,68	82,51	1,71	-34,95
UTC195342280618	2	220,12	166,82	224,28	46,41	226,49	-73,34	1,7	-175,42	1,68	63,60	1,71	-54,43
UTC195342280618	3	221,06	146,20	224,85	25,78	227,49	-93,48	1,71	164,53	1,68	44,12	1,72	-73,34
UTC195342280618	4	221,91	123,28	224,7	2,86	226,68	-115,83	1,71	142,18	1,68	21,77	1,71	-95,68
UTC195342280618	5	220,76	101,50	224,08	-19,48	225,5	-139,32	1,71	118,69	1,67	-2,29	1,71	-119,84
UTC195342280618	6	220,9	79,07	224,65	-42,40	224,99	-161,67	1,71	96,35	1,67	-24,64	1,7	-142,18
UTC195342280618	7	220,47	58,44	224,98	-63,03	225,54	177,71	1,71	75,63	1,68	-45,84	1,7	-163,38
UTC195342280618	8	220,07	38,96	225,22	-82,51	225,52	157,65	1,71	56,15	1,68	-65,32	1,7	177,14
UTC195342280618	9	219,66	17,76	225,03	-103,22	225,89	137,03	1,7	35,52	1,68	-85,94	1,7	156,51
UTC195342280618	0	219,65	-4,01	224,45	-125,00	226,32	115,26	1,7	13,18	1,68	-107,81	1,7	134,16

Observa-se que o FS (*Fracsec*) é um número que começa em 1 e vai até 0 (o zero representa o número 10). O FS identifica cada *frame* de dados em um dado segundo (são 10 *frames* por segundo). O horário e data UTC estão no formato *hora:minuto:seg:dia:mês:ano*. Desta forma, o UTC superior da Tabela 9 corresponde às 19 horas, 53 minutos e 41 segundos do dia 28 de junho de 2018. Ressalta-se que o fuso-horário é o UTC 0, que está 3 horas à frente do horário de Brasília.

Cada *frame* da Tabela 9 corresponde a uma "fotografia" do sistema observado, contendo módulo e fase de cada tensão e corrente. As formas de onda de cada grandeza medida podem ser reconstituídas a partir das informações de magnitude e fase colhidas. A Figura 49 mostra os sinais trifásicos de tensão e corrente plotados no MATLAB a partir das informações obtidas pela PMU.

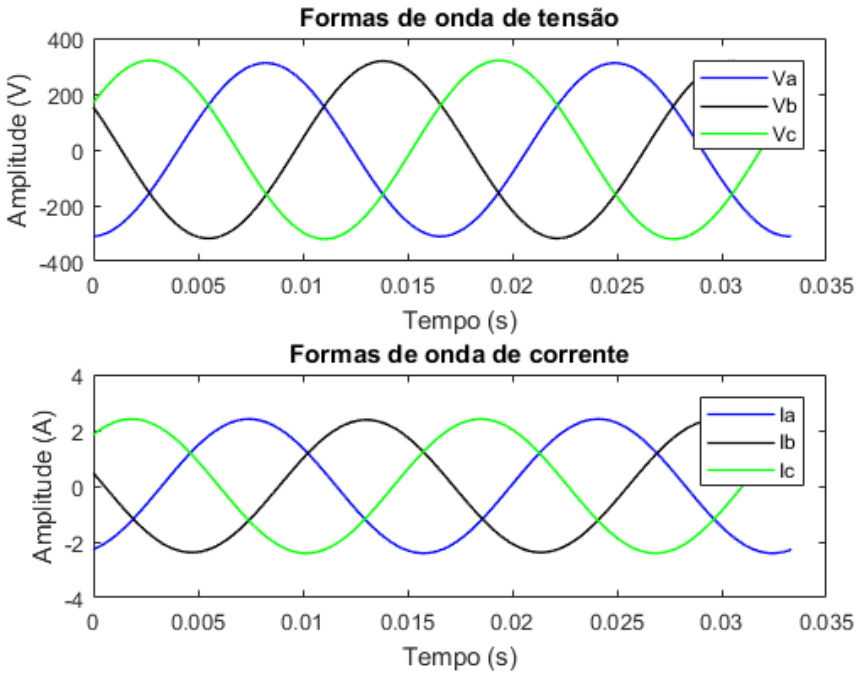

Figura 49. Sinais de tensão e corrente reconstituídos no MATLAB.

Os sinais da Figura 49 correspondem ao primeiro *frame* de dados da Tabela 9. Observa-se que a defasagem entre as três fases é de aproximadamente 120º, como esperado.

Obtém-se uma média entre as magnitudes obtidas para cada fasor e obtém-se uma média das defasagens entre os fasores. Considerando V_a como a referência angular do sistema (ângulo de fase 0º), os fasores obtidos para a tensão da rede elétrica em 220 V, a partir das médias calculadas, são apresentados na Tabela 10.

Tabela 10. Fasores obtidos pela PMU em teste.

	Va	Vb (V)	Vc (V)	Ia	Ib	Ic
Magnitude	220,21 V	224,36 V	225,82 V	1,7 A	1,67 A	1,7 A
Fase	0°	-120,96°	119,56°	17,57°	-103,42°	138,91°

A partir dos valores da Tabela 10, pode-se construir um diagrama fasorial do sistema, como mostra a Figura 50.

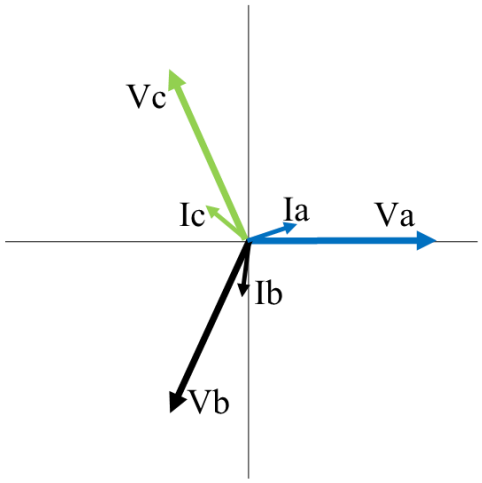

Figura 50. Diagrama fasorial do sistema trifásico de 220 V a partir dos dados colhidos pela PMU proposta.

Observa-se que apesar da carga utilizada ser resistiva, existe uma defasagem entre as tensões e correntes da mesma fase. As correntes estão adiantadas em relação às tensões em aproximadamente 15 º. A defasagem observada se deve principalmente à falta de calibragem da instrumentação utilizada.

Os valores de tensão obtidos pela PMU são bem próximos aos valores de referência da Tabela 8. Quanto aos valores de correntes obtidos pela PMU, houveram discrepâncias dos valores de magnitude em relação aos valores de referência. No entanto, estas discrepâncias não afetam qualitativamente os resultados obtidos nos testes.

5.6.2. Vários valores de tensão de entrada

A fonte trifásica utilizada foi ajustada para diferentes valores de tensão, e as medições da PMU foram realizadas. As leituras realizadas pela PMU para os valores de tensão de entrada de 230 V, 210 V e 190 V estão disponíveis no Apêndice B.

O teste em que a tensão da fonte é variada de 120 V até 0 V foi realizado. A tensão foi decrementada gradualmente em intervalos de 10 V, utilizando a manopla do varivolt trifásico. O objetivo deste teste é determinar a partir de qual tensão a PMU passa a perder a precisão de maneira considerável.

A Figura 51 mostra um gráfico da magnitude da tensão V_A obtida pela PMU para vários valores de tensão de entrada do sistema.

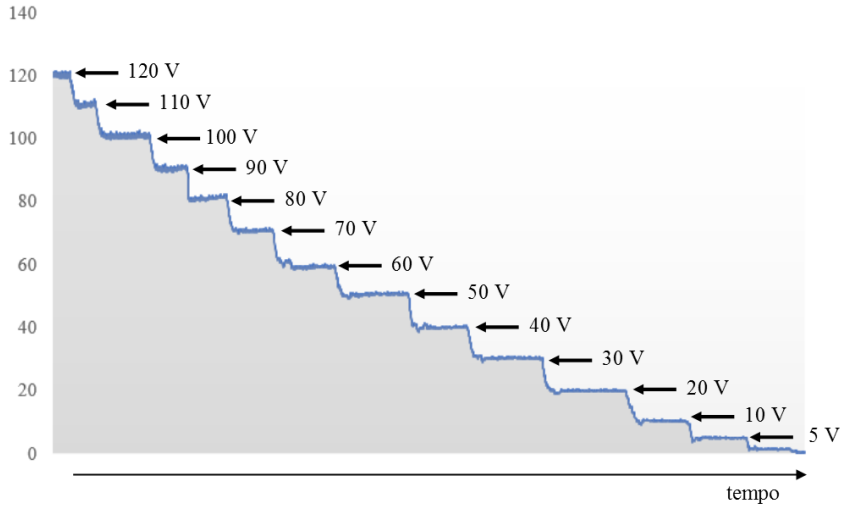

Figura 51. Magnitude de tensão lida pela PMU para uma tensão variável da fonte.

O gráfico da Figura 51 representa um monitoramento em tempo real do sistema, em que a tensão da rede elétrica monitorada sofre constantes variações.

As medições realizadas pelo protótipo foram consideradas próximas ao valor de referência para todos os valores de tensão da fonte acima de 5 V. Portanto, as medidas obtidas são consideradas coerentes para uma faixa de tensão de 5 V a 230 V.

5.6.3. Variação do ângulo de fase

A Figura 52 mostra a variação de fase obtida dos testes com tensão em 220 V.

Sabe-se que raramente a frequência da rede elétrica é exatamente 60 Hz. O desvio da frequência nominal do sistema faz com que a cada medição dos fasores da rede elétrica, a fase sofra uma variação que é proporcional ao desvio da frequência nominal do sistema.

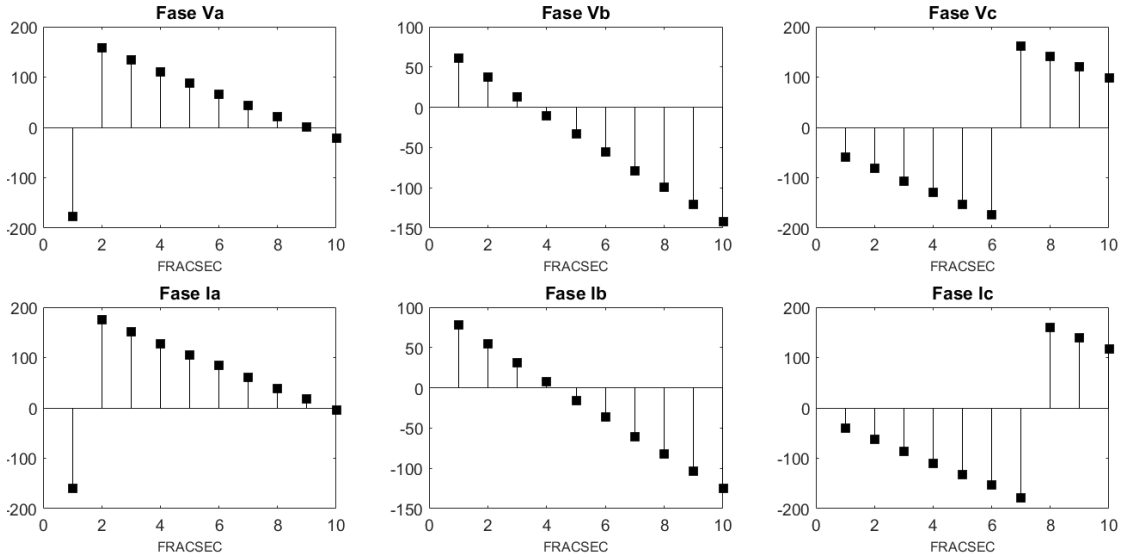

Figura 52. Variação de fase medida pelo protótipo.

Os dados da Figura 52 podem ser utilizados para estimar a frequência real da rede elétrica (IEEE, 2011). Recomenda-se o uso de um filtro digital ou modelo probabilístico para realizar estimação de frequência e ROCOF com boa precisão.

Tanto a estimação de frequência quanto a estimação de ROCOF não foram embarcadas no protótipo de PMU proposto.

5.7. Síntese

O protótipo proposto para uma PMU de baixo custo com Raspberry Pi 3 foi montado em uma *protoboard*, com os TCs e TPs, circuitos de condicionamento de sinais, filtros passivos e os dispositivos empregados. Este protótipo faz a medição fasorial sincronizada de três canais de tensão e três canais de corrente, e gera 10 *frames* de medições por segundo.

O receptor GPS utilizado é o Ublox NEO-6M-01-001 contido no módulo GPS GY-GPS6MV2. A comunicação do módulo GPS com o Raspberry Pi 3 é feita via protocolo UART.

O conversor A/D utilizado é um MCP3008 de 10 bits de resolução que suporta comunicação via SPI e o oscilador que envia os pulsos na frequência de amostragem para o Raspberry Pi 3 é o NE555.

Os testes do protótipo foram realizados mediante a medição das grandezas fasoriais de um circuito trifásico simples, com carga resistiva equilibrada. A fonte de tensão trifásica foi

ajustada em diversos valores, e as medições realizadas pelo protótipo foram enviadas a um computador em rede local (PDC).

Verificou-se que os dados colhidos são coerentes. O protótipo apresenta um bom desempenho e realiza a medição fasorial com boa precisão.

6. CONCLUSÕES

Uma proposta de PMU com Raspberry Pi 3 foi apresentada. Um circuito elétrico para a aquisição dos sinais elétricos da rede foi proposto, assim como um modelo de *software* em *multithreading* para realizar o processamento das amostras e gerar os valores fasoriais de tensão e corrente.

Um módulo GPS foi utilizado para sincronizar as leituras dos sinais da rede elétrica com o pulso por segundo (PPS) do GPS. Cada leitura realizada possui etiqueta de tempo fornecida pelo GPS.

6.1. Objetivos alcançados

O protótipo desenvolvido custa R$ 683,63 e atende aos requisitos mínimos de taxa de reporte de dados estabelecidos pela Norma IEEE C37.118.1, ou seja, o protótipo gera 10 frames por segundo, e envia os dados para um concentrador de dados via rede local. Por atender a Norma IEEE C37.118.1, pode-se afirmar que o protótipo proposto apresentou desempenho adequado, inclusive com a coleta dos fasores.

Uma fonte trifásica variável foi utilizada para fornecer tensão a um circuito trifásico equilibrado. Os fasores de tensão obtidos pela PMU apresentam defasagem de aproximadamente 120º entre as três fases em todas as medições, o que era esperado. Da mesma forma, os fasores obtidos de corrente também apresentam defasagem de 120º entre as três fases. Este resultado era esperado, já que a carga utilizada é equilibrada.

As informações fasoriais obtidas da rede permitem a reconstituição das formas de onda do sistema através do uso de alguma ferramenta gráfica. O fato do protótipo fornecer 10 frames de dados por segundo significa que o monitoramento da rede elétrica ocorre em tempo real. Os dados obtidos possuem etiqueta de tempo, o que permite ao usuário ter conhecimento do momento exato em que ocorrem mudanças no estado de operação do sistema.

A sincronização das leituras com o GPS permite que várias PMUs ao longo de uma grande área geográfica possam realizar leituras simultâneas da rede, permitindo que o operador do sistema possa ter uma visão completa do estado da rede elétrica com referência de tempo.

A análise dos dados obtidos também permite a estimação da frequência do sistema. Recomenda-se o uso de algum modelo de filtro digital ou modelo probabilístico para gerar as estimações de frequência a partir da amostragem do sinal.

6.2. Limitações

O protótipo proposto não realiza a estimação de frequência embarcada, como é especificado pela Norma IEEE C37.118.1. Esta estimação pode ser realizada a partir dos dados de variação do ângulo de fase de cada fasor medido.

A janela de amostragem do protótipo desenvolvido não é uniforme ao longo de cada segundo. O protótipo realiza a aquisição dos dados durante 833,33 ms, e passa 167,67 ms sem colher dados (tempo morto).

O fato do Raspberry Pi 3 ser um computador que possui um sistema operacional que não é de tempo real, dificulta a tarefa de determinar o momento exato que cada operação deve ocorrer. Desta forma, ajustar o tempo em que ocorre a amostragem e a geração dos *frames* de dados de maneira uniforme no Raspberry Pi 3 é uma tarefa difícil e que requer conhecimentos mais profundos sobre sistemas operacionais de tempo real.

6.3. Trabalhos futuros

Considera-se que a discussão proposta neste trabalho sobre o desenvolvimento de uma PMU com o Raspberry Pi 3 pode ser ampliada, de modo que se consiga alcançar todos os requisitos da Norma IEEE C37.118.1 para PMUs.

As recomendações para realização de trabalhos futuros sobre o tema discutido podem ser listadas como segue:

- Embarcar a estimação de frequência ao protótipo;
- Propor um algoritmo probabilístico e/ou um filtro digital para o processamento da frequência e ROCOF, de modo a diminuir a imprecisão dos resultados devido aos ruídos;
- Melhorar o protótipo proposto. Fornecer opções ao usuário, tais como: número de canais a serem utilizados, número de *frames* por segundo gerados, frequência de amostragem utilizada, entre outras;
- Realizar um estudo mais aprofundado sobre o funcionamento do Raspberry Pi 3, e estudar alternativas de sistemas operacionais de tempo para o Raspberry Pi 3.

REFERÊNCIAS

ALEIXO, Renato Ribeiro. **Proposta e Implementação de uma Micro-PMU.** 2018. 102 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal de Juiz de Fora, Juiz de Fora, Mg, Brasil, 2018.

ALI, Syed Akif et al. Real time implementation of non-DFT based three phase phasor measurement unit as per IEEE standard C37.118.1. **2017 IEEE International Conference On Environment And Electrical Engineering And 2017 IEEE Industrial And Commercial Power Systems Europe (EEEIC / I&CPS Europe)**, [s.l.], jun. 2017. IEEE.

ANISHA, N. P.; PRASANNA, D.; SUYAMPULINGAM, A. Soft Phasor Measurement Unit. **In Procedia Technology**, 21, 2015. 533-539. Disponível em: <http://www.sciencedirect.com/science/article/pii/S221201731500273X>. Acesso em: outubro 2017.

BARNEY, Blaise. Introduction to Parallel Computing. Livermore National Laboratory. Disponível em: < https://computing.llnl.gov/tutorials/parallel_comp/>. Acesso em: 28 mai. 2018.

CALTELLO, Paolo; MUSCAS, Carlo; PEGORARO, Paolo A. Adaptive Management Of Synchrophasor Latency For An Active Phasor Data Concentrator. **2017 IEEE International Instrumentation and Measurement Technology Conference**. Torino, Italia. p. 1-6. maio 2017.

DENNIS, Andrew K. **Raspberry Pi Computer Architecture Essentials**. Birmingham, UK: Packt Publishing, 2016.

FANG, X. et al. Smart Grid - The New and Improved Power Grid: A Survey. **IEEE Communications Surveys and Tutorials**, 2012. 944-980.

FERREIRA, Matheus Ribeiro. **Desenvolvimento De Um Protótipo De Sistema Microprocessado Capaz De Realizar Leituras Sincronizadas De Uma Pmu.** 2017. 60 f. Monografia (Especialização) - Curso de Engenharia Elétrica, Universidade Federal do Maranhão, São Luís, 2017.

FREE SOFTWARE FOUNDATION. **GCC, the GNU Compiler Collection**. EUA, 2018. Disponível em: <https://gcc.gnu.org/>. Acesso em: 26 jun. 2018.

GAY, Warren. **Mastering the Raspberry Pi**. New York, EUA: Springer Science, 2014. 482 p.

GRAMA, Ananth et al. **Introduction to Parallel Computing.** 2. ed. EUA: Pearson Education, 2003. 856 p.

IEEE Standard Association. IEEE Std C37.118.1. IEEE Standard for Synchrophasor Measurements for Power Systems. Nova York, NY, EUA, 2011.

IEEE Standard Association. IEEE Std C37.118.1. IEEE Standard for Synchrophasor Measurements for Power Systems. Amendment 1: Modification of Selected Performance Requirements. Nova York, NY, EUA, 2014.

IEEE Standard Association. IEEE Std C37.118.2. IEEE Standard for Synchrophasor Data Transfer for Power Systems. Nova York, NY, EUA, 2011.

LI, Qing; YAO, Carolyn. **Real-Time Concepts for Embedded Systems**. San Francisco, EUA: CMP Books, 2003.

LIMA, Amanda de Aguiar Serra. **Estudo Teórico e Experimental da Medição Fasorial Sincronizada.** 2015. 83 f. Monografia (Especialização) - Curso de Engenharia Elétrica, Universidade Federal do Maranhão, São Luís, 2015.

LOPES, Denivaldo. **Sistemas Operacionais de Tempo Real.** São Luís, MA: Universidade Federal do Maranhão, 2009. 356 slides, color.

MICROCHIP. MCP3008 datasheet. EUA, 2007. Disponível em: < http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf>. Acesso em: jun. 2018.

MONTI, Antonello; PONCI, Ferdinanda. Electric Power Systems. **Intelligent Monitoring, Control, And Security Of Critical Infrastructure Systems**, Berlim, Alemanha, p.31-65, 14 set. 2014. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-662-44160-2_2.

MONTICELLI, Alcir José. **Fluxo de Carga em Redes de Energia Elétrica**. São Paulo, SP: Edgard Blücher, 1983.

NALON, José A. **Introdução ao Processamento Digital de Sinais**. Rio de Janeiro, RJ: LTC, 2009.

NILSSON, James W.; RIEDEL, Susan A. **Circuitos Elétricos.** 8. ed. São Paulo, SP: Pearson Prentice Hall, 2009.

NMEA. **NMEA 0183 Standard**. Disponível em: <https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp>. Acesso em: 01 jul. 2018.

PHADKE, A. G.; THORP, J. S. **Synchronized Phasor Measurement and Their Applications.** New York, Eua: Springer, 2008. 254 p.

RASPBERRY PI FOUNDATION. Raspberry Pi: Teach, Learn, and Make With Raspberry Pi. Disponível em: < https://www.raspberrypi.org/>. Acesso em: 10 jun. 2018.

ROBERTS, Stephen. **Lecture 7 – The Discrete Fourier Transform**. Material de Aula. 2003. Disponível em: < http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf>. Acesso em: 26 jun. 2018.

STALLMAN, Richard M. **Using the GNU Compiler Collection:** For GCC version 9.0.0. Boston, EUA: GNU Press, 2003. 950 p. Disponível em: <https://gcc.gnu.org/onlinedocs/gcc.pdf>. Acesso em: 09 jun. 2018.

ST MICROELETRONICS. NE555 datasheet. 2012. Disponível em: <https://www.st.com/resource/en/datasheet/cd00000479.pdf>. Acesso em: jun. 2018.

TANENBAUM, Andrew S.; WETHERALL, David J. **Computer Networks**. 5. ed. EUA: Pearson Prentice Hall, 2010.

THORP, J. S.; PHADKE, A. G. **Computer Relaying in Power Systems.** Pensilvânia, EUA: C.T. Leondes, 1991. 57 p.

THE PI4J PROJECT. **Pin Numbering – Raspberry Pi 3 Model B.** Disponível em: <http://pi4j.com/pins/model-3b-rev1.html>. Acesso em: 28 jun. 2018.

U.S. DEPARTMENT OF ENERGY. Factors Affecting PMU Installation Costs. Smart Grid Investment Grant Program. Setembro de 2014. Disponível em < https://www.smartgrid.gov/files/PMU-cost-study-final-10162014_1.pdf>. Acesso em abril de 2018.

VAIL, Jonathan *et al*. Global Positioning System. US Environmental Protection Agency. Athens, Geógia, EUA. 23 jun. 2015.

WILLISTON, Kenton. **Digital Signal Processing**. Burlington, EUA: Elsevier, 2009.

WIRING PI. **WiringPi.** 2018. Disponível em: <http://wiringpi.com/>. Acesso em: 09 jun. 2018.

ZHANG, Yingchen et al. Wide-Area Frequency Monitoring Network (FNET): Architecture and Applications. **IEEE Transactions On Smart Grid.** Argone, EUA, p. 159-167. 08 jul. 2010.

APÊNDICE A – CÓDIGOS UTILIZADOS PARA COMUNICAÇÕES

A.1 CÓDIGO EM C PARA COMUNICAÇÃO SPI COM O MCP3008

//Biblioteca de funções utilizadas pelo MCP3008

```
/******************** INCLUIR BIBLIOTECAS ***************************/
#include <fcntl.h> //Needed for SPI port
#include<sys/ioctl.h> //Needed for SPI port
#include <linux/spi/spidev.h> //Needed for SPI port
#include <unistd.h> //Needed for SPI port
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <linux/i2c-dev.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <wiringPi.h>
/******************** FIM DE INCLUSAO DE BIBLIOTECAS *****************/

/******************** DECLARACAO DE VARIAVEIS ************************/
int spi_cs0_fd;                 // File descriptor for the SPI device 0
int spi_cs1_fd;                 // File descriptor for the SPI device 1
unsigned char spi_mode;         // ************************
unsigned char spi_bitsPerWord;  // Needed for SPI
unsigned int spi_speed;         // ************************
char *SPIcs1 = "/dev/spidev0.1", *SPIcs0 = "/dev/spidev0.0"; // Name for SPI ports
unsigned char data[3]; // Data acquired from SPI communication
struct spi_ioc_transfer spi[3]; // Struct of SPI elements, works as data buffer
/******************** FIM DE DECLARACAO DE VARIAVEIS ************************/

/******************** DECLARACAO DE FUNCOES ************************/
int SpiOpenPort (int spi_device);
int SpiClosePort (int spi_device);
void setupSpiWR(int spi_device, int length);
int mcp3008WR(int spi_device, int channel);
/******************** FIM DE DECLARACAO DE FUNCOES ***************/

//Funcao que a comunicacao SPI com o MCP3008
int SpiOpenPort (int spi_device){
      int status_value = -1; // Initiated as closed
      int *spi_cs_fd; // File descriptor receiver
      spi_mode = SPI_MODE_0; // Clock idle low, data is clocked in on rising
                        //edge, output data (change) on falling edge
      spi_bitsPerWord = 8; // Set bits per word
      spi_speed = 1000000;   //mSet bus speed. 1000000 = 1MHz (1uS per bit)
      if(spi_device){ // For CS1
        spi_cs_fd = &spi_cs1_fd;
      }
      else{ // For CS0
        spi_cs_fd = &spi_cs0_fd;
      }
      if(spi_device){ // For CS1
        *spi_cs_fd = open(SPIcs1, O_RDWR); // Open port for reading and writing
      }
      else{ // For CS0
        *spi_cs_fd = open(SPIcs0, O_RDWR); // Open port for reading and writing
```

```
    }
    if(*spi_cs_fd < 0){
       perror("Error - Could not open SPI device");
       exit(1);
    }

    // Set the port option for writing and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_WR_MODE, &spi_mode);
    if(status_value < 0){
         perror("Could not set SPIMode (WR)...ioctl fail");
       exit(1);
    }
    // Set the port option for reading and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_RD_MODE, &spi_mode);
    if(status_value < 0){
        perror("Could not set SPIMode (RD)...ioctl fail");
        exit(1);
    }

    // Set port option for bits per word at writing operation and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_WR_BITS_PER_WORD, &spi_bitsPerWord);
    if(status_value < 0){
        perror("Could not set SPI bitsPerWord (WR)...ioctl fail");
        exit(1);
    }

    // Set port option for bits per word at reading operation and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_RD_BITS_PER_WORD, &spi_bitsPerWord);
    if(status_value < 0){
        perror("Could not set SPI bitsPerWord(RD)...ioctl fail");
        exit(1);
    }

    // Set port option for max speed at writing operation and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_WR_MAX_SPEED_HZ, &spi_speed);
    if(status_value < 0){
        perror("Could not set SPI speed (WR)...ioctl fail");
        exit(1);
    }
    // Set port option for bits per word at reading operation and the address
    status_value = ioctl(*spi_cs_fd, SPI_IOC_RD_MAX_SPEED_HZ, &spi_speed);
    if(status_value < 0){
        perror("Could not set SPI speed (RD)...ioctl fail");
        exit(1);
    }
    return(status_value);
}
// Close the SPI bus communication (spi_device 0=CS0, 1=CS1)
int SpiClosePort (int spi_device){
   int status_value = -1; // Initiated as closed
   int *spi_cs_fd; // File descriptor receiver
   if(spi_device){ // For CS1
     spi_cs_fd = &spi_cs1_fd;
   }
   else{ // For CS0
     spi_cs_fd = &spi_cs0_fd;
   }
   status_value = close(*spi_cs_fd); // Close the SPI communication
   if(status_value < 0){
     perror("Error - Could not close SPI device");
```

```
      exit(1);
    }
   return(status_value);
}

//Configurar a interface SPI
void setupSpiWR(int spi_device, int length){
      int i = 0; // For use on loops


  for(i = 0 ; i < length ; i++){ // One SPI transfer for each byte
    memset (&spi[i], 0, sizeof (spi[i])); // Struct needs to be initialised to NULL
    spi[i].tx_buf = (unsigned long)(data + i); // Transmit from 'data'
    spi[i].rx_buf = (unsigned long)(data + i) ; // Receive into 'data'
    spi[i].len = sizeof(*(data + i)) ; //
    spi[i].delay_usecs = 0 ; // Specifies the parameters
    spi[i].speed_hz = spi_speed ; // in order to the SPI struct
    spi[i].bits_per_word = spi_bitsPerWord ; // to work
    spi[i].cs_change = 0; //
  }

}

//Realiza toda a configuracao do MCP3008
void mcp3008_open(int spi_device){
  SpiOpenPort(spi_device);
  setupSpiWR(spi_device,3);
}

//Encerrar a comunicacao com o MCP3008
void mcp3008_close(int spi_device){
  SpiClosePort(spi_device);
}
//Retorna a leitura do MCP
int mcp3008WR(int spi_device, int channel){
  int ADRaw; // Holds raw data read from ADC channel
  int retVal = -1; // Initiated as closed
  int *spi_cs_fd; // File descriptor receiver
      if(spi_device){ // For CS1
        spi_cs_fd = &spi_cs1_fd;
      }
      else{ // For CS0
        spi_cs_fd = &spi_cs0_fd;
      }
      data[0] = 1; // First byte transmitted -> start bit
      // Second byte transmitted -> (SGL/DIF = 1, D2=D1=D0=0)
      data[1] = 0b10000000 |( ((channel & 7) << 4));
      data[2] = 0; // Third byte transmitted -> don't care
  retVal = ioctl(*spi_cs_fd, SPI_IOC_MESSAGE(3), &spi) ; // Performs the write-read of the SPI struct
  if(retVal < 0){
    perror("Error - Problem transmitting spi data..ioctl");
    exit(1);
  }
      ADRaw = 0; // Resets for each loop interaction
      ADRaw = (data[1]<< 8) & 0b1100000000; // Merge data[1] & data[2]
      ADRaw |= (data[2] & 0xff); // to get result
      return ADRaw;
}
```

A.2 CÓDIGO EM C PARA COMUNICAÇÃO UART COM O GY-GPS6MV2

```
//Obter a informacao de UTC do modulo GPS
void get_utc(char*UTC){
	int uart0_filestream = -1;
	int flag = 0;
	int cont2 = 0;
	int cont = 0;
	struct termios options;
	char alvo[8] = "RMC";
	uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);		//Open
in non blocking read/write mode

	if (uart0_filestream == -1)
	{
		//ERROR - CAN'T OPEN SERIAL PORT
		printf("Error - Unable to open UART.  Ensure it is not in use by another application\n");
	}else printf("Conection opened sucessfully!\n");

	//CONFIGURE THE UART
	tcgetattr(uart0_filestream, &options);
	options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;		//<Set baud rate
	options.c_iflag = IGNPAR;
	options.c_oflag = 0;
	options.c_lflag = 0;
	tcflush(uart0_filestream, TCIFLUSH);
	tcsetattr(uart0_filestream, TCSANOW, &options);

	printf("Getting UTC reference...\n");
	while(1){
		if (uart0_filestream != -1)
		{
			// Read up to 255 characters from the port if they are there
			unsigned char rx_buffer[256];
			int rx_length = read(uart0_filestream, (void*)rx_buffer, 255);		//Filestream,
buffer to store in, number of bytes to read (max)
			if (rx_length < 0)
			{
				//An error occured (will occur if there are no bytes)
			}
			else if (rx_length == 0)
			{
				//No data waiting
			}
			else
			{
				//Bytes received
				rx_buffer[rx_length] = '\0';
				if(flag){
					cont++;
					if(cont==1){
						UTC[1] = rx_buffer[0];
						UTC[2] = rx_buffer[1];
						UTC[3] = rx_buffer[2];
						UTC[4] = rx_buffer[3];
						UTC[5] = rx_buffer[4];

					}
					if(cont == 6){
						UTC[6] = rx_buffer[5];
```

```
                                        UTC[7] = rx_buffer[6];
                                        UTC[8] = rx_buffer[7];
                                }
                                if(cont == 7){
                                        UTC[9] = rx_buffer[0];
                                        UTC[10] = rx_buffer[1];
                                        UTC[11] = rx_buffer[2];
                                        UTC[12] = '\0';
                                        printf("UTC: %s\n",UTC);
                                        break;
                                }
                        }
                        if(strstr(rx_buffer,alvo)!=NULL){
                                //printf("RMC LOCATED!\n");
                                UTC[0] = rx_buffer[7];
                                flag = 1;
                                if(++cont2==50) break;
                        }
                }
        }
    }

    //----- CLOSE THE UART -----
    close(uart0_filestream);
}
```

A.3 CÓDIGO EM C PARA COMUNICAÇÃO UDP

```
//Este arquivo contem funcoes responsaveis pela conexao
//do prototipo via rede
/***************INCLUSAO DE BIBLIOTECAS ******************/
#include<stdio.h> //printf
#include<string.h> //memset
#include<stdlib.h> //exit(0);
#include<arpa/inet.h>
#include<sys/socket.h>
/*********************************************************/
/*************** DEFINICOES ******************************/
#define BUFLEN 512  //Max length of buffer
/*********************************************************/

/************ DECLARANDO VARIAVEIS **************/
   struct sockaddr_in si_other;
   int s, slen=sizeof(si_other);
   char buf[BUFLEN];
/************************************************/

//Funcao que eh chamada em caso de erro
void die(char *s)
{
   perror(s);
   exit(1);
}

//Esta funcao cria um socket UDP e configura
//o endereco do servidor
void create_udp_socket(int port, const char* server_ip){
        /******* CRIANDO E CONFIGURANDO UM SOCKET *******/
   if ( (s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
```

```
    {
        die("socket");
    }

    memset((char *) &si_other, 0, sizeof(si_other));
    si_other.sin_family = AF_INET;
    si_other.sin_port = htons(port);
    /*************************************************/

    //Convertendo a string do enderenco de rede em um endereco
    //de rede valido
    if (inet_aton(server_ip , &si_other.sin_addr) == 0)
    {
        fprintf(stderr, "inet_aton() failed\n");
        exit(1);
    }
}

//Envia um pacote udp para o servidor informado na funcao "create_udp_socket"
void send_udp_packet(const char*message){
            //send the message
        if (sendto(s, message, strlen(message) , 0 , (struct sockaddr *) &si_other, slen)==-1)
        {
            die("sendto()");
        }
}

//Recebe um pacote UDP do servidor configurado anteriormente
// em "create_udp_socket"
char* receive_udp_packet(void){
            //receive a reply and print it
    //clear the buffer by filling null, it might have previously received data
    memset(buf,'\0', BUFLEN);
    //try to receive some data, this is a blocking call
    if (recvfrom(s, buf, BUFLEN, 0, (struct sockaddr *) &si_other, &slen) == -1)
    {
        die("recvfrom()");
    }
    return buf;
}
```

APÊNDICE B – DADOS GERADOS PELA PMU

B.1 TESTE COM 230 V

Tabela 11. Medições realizadas pelo protótipo da PMU para 230 V.

Horário/Data UTC	FS	Mag Va	Fase Va	Mag Vb	Fase Vb	Mag Vc	Fase Vc	Mag Ia	Fase Ia	Mag Ib	Fase Ib	Mag Ic	Fase Ic
UTC200512280618	1	232,08	-77,92	232,17	159,95	234,45	40,68	1,76	-61,88	1,71	175,99	1,75	58,44
UTC200512280618	2	231,99	-96,92	232,43	140,47	234,16	21,20	1,76	-81,36	1,71	156,51	1,75	38,96
UTC200512280618	3	231,65	-116,40	232,78	120,99	234,01	1,15	1,76	-100,93	1,71	137,03	1,74	19,48
UTC200512280618	4	231,25	-136,46	232,91	100,93	234,13	-18,91	1,76	-120,99	1,72	116,97	1,74	-0,57
UTC200512280618	5	230,83	-157,08	233,01	80,79	234,52	-38,96	1,76	-141,04	1,72	96,92	1,74	-21,20
UTC200512280618	6	230,86	-176,56	232,58	61,31	234,98	-59,01	1,76	-161,09	1,72	77,35	1,75	-40,68
UTC200512280618	7	230,98	162,81	232,13	40,68	235,20	-79,07	1,75	178,85	1,71	57,30	1,75	-60,73
UTC200512280618	8	231,36	143,33	231,90	21,20	235,19	-98,64	1,76	158,80	1,71	37,24	1,75	-80,79
UTC200512280618	9	233,19	123,85	233,39	2,29	236,75	-116,97	1,77	141,04	1,72	20,05	1,76	-97,49
UTC200512280618	0	232,71	102,65	232,71	-19,48	234,16	-138,75	1,77	119,27	1,71	-2,29	1,75	-119,84
UTC200513280618	1	232,00	89,38	232,68	-33,23	234,56	-152,50	1,76	104,94	1,71	-16,62	1,75	-134,74
UTC200513280618	2	231,73	71,05	232,78	-51,57	234,16	-170,83	1,76	86,61	1,71	-34,95	1,75	-152,50
UTC200513280618	3	231,36	53,29	232,97	-69,33	234,42	171,41	1,76	69,33	1,71	-53,29	1,75	-170,83
UTC200513280618	4	231,23	35,52	232,95	-87,09	234,59	153,07	1,76	50,99	1,72	-71,05	1,74	171,41
UTC200513280618	5	231,04	17,19	232,82	-104,94	234,92	134,74	1,76	32,66	1,72	-89,38	1,75	153,07
UTC200513280618	6	232,05	-0,57	232,71	-122,13	235,39	118,12	1,76	16,04	1,71	-105,52	1,75	136,46
UTC200513280618	7	233,03	-17,76	233,92	-139,32	236,82	100,93	1,77	-0,57	1,73	-121,56	1,76	120,99
UTC200513280618	8	231,20	-37,82	232,17	-159,37	235,16	80,79	1,76	-21,77	1,71	-143,33	1,75	98,64
UTC200513280618	9	231,70	-57,30	231,82	-178,85	234,93	61,31	1,76	-41,25	1,71	-162,81	1,75	79,07
UTC200513280618	0	232,21	-77,35	231,92	160,52	234,47	41,25	1,76	-61,31	1,71	177,14	1,75	59,59
UTC200514280618	1	231,75	-91,19	232,53	146,77	234,58	26,93	1,76	-75,06	1,71	162,81	1,75	45,26
UTC200514280618	2	231,86	-108,38	232,63	129,58	234,43	9,74	1,76	-92,25	1,71	145,62	1,75	28,07
UTC200514280618	3	231,27	-126,71	232,95	111,25	234,37	-8,59	1,76	-110,67	1,71	127,29	1,75	9,74
UTC200514280618	4	232,30	-144,48	234,10	94,06	235,67	-25,78	1,77	-127,29	1,72	111,25	1,75	-6,30
UTC200514280618	5	232,37	-162,81	233,80	75,63	235,95	-43,54	1,77	-145,62	1,72	92,91	1,75	-24,64
UTC200514280618	6	231,35	-179,43	232,78	58,44	235,32	-61,31	1,76	-163,38	1,72	74,48	1,75	-42,97
UTC200514280618	7	231,43	163,96	232,47	42,40	235,20	-77,92	1,76	180,00	1,72	58,44	1,75	-59,59
UTC200514280618	8	231,66	147,91	232,37	25,78	235,23	-94,06	1,76	163,96	1,71	41,83	1,75	-76,20
UTC200514280618	9	231,53	131,87	232,34	9,74	235,16	-110,10	1,76	147,91	1,71	25,78	1,75	-91,67
UTC200514280618	0	231,96	114,11	232,23	-8,02	234,83	-127,29	1,76	130,15	1,71	8,59	1,75	-109,53
UTC200515280618	1	231,81	83,079	232,71	-38,961	234,83	-158,8	1,76	99,213	1,71	-22,918	1,75	-140,47
UTC200515280618	2	233,63	68,182	234,13	-53,858	236,43	-172,55	1,77	85,3707	1,72	-36,096	1,76	-153,07
UTC200515280618	3	231,8	50,993	233,61	-71,047	234,52	169,11	1,76	67,609	1,72	-54,431	1,75	-171,98
UTC200515280618	4	231,58	34,377	232,85	-87,663	234,76	152,5	1,76	50,4203	1,72	-71,62	1,75	170,83
UTC200515280618	5	231,2	17,189	232,92	-104,94	234,74	134,74	1,76	33,2316	1,72	-88,808	1,74	153,07
UTC200515280618	6	231,31	-1,1459	232,56	-123,28	235,11	116,97	1,76	14,8969	1,72	-107,23	1,75	135,31
UTC200515280618	7	231,35	-18,335	232,34	-140,47	235,17	99,786	1,76	-2,2918	1,72	-124,42	1,75	117,55
UTC200515280618	8	231,63	-36,669	232,23	-158,8	235,12	81,933	1,76	-20,626	1,71	-142,18	1,75	99,786
UTC200515280618	9	231,74	-54,431	232,14	-176,56	235,14	64,171	1,76	-38,388	1,71	-159,95	1,75	81,933
UTC200515280618	0	231,76	-71,62	232,19	166,249	234,65	46,41	1,76	-55,577	1,71	-177,71	1,75	64,744

B.2 TESTE COM 210 V

Tabela 12. Medições realizadas pelo protótipo da PMU para 210 V.

Horário/Data UTC	FS	Mag Va	Fase Va	Mag Vb	Fase Vb	Mag Vc	Fase Vc	Mag Ia	Fase Ia	Mag Ib	Fase Ib	Mag Ic	Fase Ic
UTC193811280618	1	211,24	-70,47	208,92	167,39	211,99	48,13	1,66	-52,14	1,60	-173,12	1,65	68,75
UTC193811280618	2	211,60	-102,65	210,19	134,74	211,11	16,04	1,67	-83,08	1,60	154,79	1,64	37,82
UTC193811280618	3	208,85	-135,31	210,06	102,08	210,48	-17,76	1,66	-116,97	1,60	120,99	1,62	2,86
UTC193811280618	4	207,95	-168,54	209,62	69,33	211,68	-50,99	1,64	-150,21	1,61	87,75	1,63	-30,94
UTC193811280618	5	208,37	157,65	207,64	36,10	212,46	-84,22	1,64	175,99	1,60	54,43	1,64	-64,17
UTC193811280618	6	210,11	125,00	207,87	2,86	212,01	-116,40	1,65	143,33	1,60	21,77	1,64	-96,35
UTC193811280618	7	210,74	93,48	208,86	-29,22	210,72	-148,49	1,66	111,25	1,59	-10,31	1,64	-128,43
UTC193811280618	8	209,77	61,88	209,84	-61,31	210,17	179,43	1,66	79,64	1,60	-42,40	1,63	-160,52
UTC193811280618	9	208,60	29,79	210,13	-92,91	211,19	147,34	1,65	48,13	1,61	-73,91	1,63	167,39
UTC193811280618	0	208,53	-1,15	209,21	-123,28	212,48	116,40	1,64	17,19	1,61	-104,94	1,63	136,46
UTC193812280618	1	210,66	-80,21	208,62	157,08	210,85	37,82	1,65	-61,88	1,59	175,99	1,64	58,44
UTC193812280618	2	210,25	-110,67	209,71	126,71	210,51	6,88	1,66	-92,25	1,60	145,05	1,63	27,50
UTC193812280618	3	208,54	-142,18	210,10	95,20	210,82	-24,64	1,65	56,15	1,60	113,54	1,62	-4,58
UTC193812280618	4	208,40	-174,27	209,55	64,17	212,11	-56,72	1,64	24,64	1,61	82,51	1,63	-36,10
UTC193812280618	5	209,14	154,79	208,14	32,66	212,65	-87,09	1,64	-6,88	1,60	51,57	1,64	-67,61
UTC193812280618	6	210,18	123,28	207,93	1,15	212,02	-118,69	1,65	-38,96	1,59	20,05	1,64	-98,64
UTC193812280618	7	210,69	91,19	208,75	-31,51	210,50	-150,78	1,66	-70,47	1,59	-12,61	1,64	-130,15
UTC193812280618	8	209,78	60,16	209,83	-63,03	210,37	177,71	1,66	78,50	1,60	-44,12	1,63	-162,24
UTC193812280618	9	210,49	29,79	211,60	-92,34	212,81	148,49	1,67	49,85	1,61	-72,19	1,64	170,26
UTC193812280618	0	208,82	-2,29	209,55	-123,85	212,50	115,26	1,64	16,62	1,61	-104,94	1,63	135,88
UTC193813280618	1	210,95	-89,38	209,14	148,49	210,80	28,65	1,66	-71,05	1,59	167,39	1,64	49,27
UTC193813280618	2	209,62	-119,84	210,02	116,97	210,46	-2,29	1,66	-101,50	1,60	135,88	1,63	17,76
UTC193813280618	3	208,47	-152,50	210,20	85,37	211,15	-34,95	1,65	-134,16	1,61	103,80	1,62	-14,90
UTC193813280618	4	208,79	175,99	209,24	53,86	212,65	-66,46	1,64	-165,68	1,61	72,77	1,64	-46,41
UTC193813280618	5	209,63	145,05	208,16	22,92	212,71	-96,92	1,65	163,38	1,60	41,83	1,64	-76,78
UTC193813280618	6	211,30	114,68	209,04	-7,45	212,38	-126,71	1,66	133,59	1,60	12,61	1,65	-105,52
UTC193813280618	7	211,97	83,65	210,54	-38,39	211,49	-157,08	1,68	103,22	1,60	-18,33	1,64	-135,31
UTC193813280618	8	209,20	50,42	210,21	-72,77	210,28	167,39	1,66	68,75	1,60	-53,86	1,63	-171,98
UTC193813280618	9	208,44	17,19	210,03	-104,94	211,74	134,74	1,65	35,52	1,61	-86,52	1,63	154,79
UTC193813280618	0	208,93	-14,90	208,79	-137,03	212,82	102,65	1,64	3,44	1,61	-118,12	1,64	122,70
UTC193814280618	1	210,46	-103,22	209,46	134,16	210,33	14,90	1,66	-84,80	1,59	153,07	1,63	34,95
UTC193814280618	2	209,06	-134,74	210,18	102,08	210,70	-17,76	1,66	-116,40	1,60	120,99	1,62	2,86
UTC193814280618	3	208,00	-168,54	209,43	69,33	211,87	-50,99	1,64	-150,21	1,61	87,75	1,63	-30,94
UTC193814280618	4	210,37	158,23	209,16	36,67	213,83	-83,08	1,65	177,71	1,61	56,72	1,65	-61,88
UTC193814280618	5	210,55	123,85	208,02	1,72	212,28	-117,55	1,65	142,76	1,59	21,77	1,64	-96,35
UTC193814280618	6	210,73	89,95	208,73	-32,66	210,48	-151,93	1,66	107,81	1,59	-13,75	1,64	-131,87
UTC193814280618	7	209,90	57,87	210,18	-64,74	210,64	175,42	1,66	76,20	1,60	-46,41	1,63	-163,96
UTC193814280618	8	208,28	25,78	210,21	-96,92	211,46	142,76	1,65	44,12	1,61	-78,50	1,63	163,38
UTC193814280618	9	208,41	-6,30	208,98	-128,43	212,57	111,25	1,64	12,03	1,61	70,47	1,64	131,30
UTC193814280618	0	209,68	-38,96	208,09	-161,09	212,81	79,07	1,64	-20,63	1,60	37,82	1,64	99,21

B.3 TESTE COM 190 V

Tabela 13. Medições realizadas pelo protótipo da PMU para 190 V.

Horário/Data UTC	FS	Mag Va	Fase Va	Mag Vb	Fase Vb	Mag Vc	Fase Vc	Mag Ia	Fase Ia	Mag Ib	Fase Ib	Mag Ic	Fase Ic
UTC201119280618	1	190,31	39,53	195,3	-81,93	194,55	158,23	1,57	59,01	1,54	-62,45	1,56	180,00
UTC201119280618	2	189,99	15,47	194,99	-105,52	195,01	135,31	1,56	35,52	1,54	-85,94	1,56	156,51
UTC201119280618	3	190,06	-8,02	194,43	-129,01	195,52	111,25	1,56	11,46	1,54	-109,53	1,56	132,44
UTC201119280618	4	190,28	-32,66	193,72	-153,07	195,55	87,09	1,56	-12,61	1,54	-133,59	1,56	108,38
UTC201119280618	5	192,17	-56,15	195,23	-176,56	196,55	64,74	1,58	-34,95	1,54	-155,36	1,57	87,18
UTC201119280618	6	192,02	-81,93	194,3	157,08	194,66	38,39	1,56	-61,88	1,53	177,71	1,57	60,16
UTC201119280618	7	191,14	-106,09	194,68	132,44	194,35	13,18	1,57	-86,52	1,53	151,93	1,56	34,38
UTC201119280618	8	190,48	-130,15	195,15	108,38	194,48	-10,89	1,57	-110,67	1,54	127,86	1,55	10,31
UTC201119280618	9	190,28	-153,07	195,16	85,37	194,97	-33,80	1,56	-133,59	1,54	104,94	1,55	-12,61
UTC201119280618	0	190,18	-175,42	194,92	63,03	195,46	-56,15	1,56	-155,94	1,54	82,51	1,56	-34,95
UTC201120280618	1	190,56	35,52	195,26	-85,37	194,75	154,79	1,57	55,58	1,54	-65,89	1,56	176,56
UTC201120280618	2	190,51	13,18	195,21	-107,81	195,97	133,02	1,56	33,23	1,54	-87,66	1,56	154,79
UTC201120280618	3	191,39	-10,89	195,67	-130,73	196,69	110,10	1,57	10,89	1,55	-110,10	1,57	132,44
UTC201120280618	4	190,71	-35,52	193,95	-155,94	195,62	84,22	1,56	-15,47	1,54	-136,46	1,56	105,52
UTC201120280618	5	191,12	-59,01	193,88	180,00	195,37	61,31	1,56	-39,53	1,53	-159,95	1,57	81,93
UTC201120280618	6	191,17	-81,36	194,34	157,08	194,83	38,39	1,57	-61,88	1,53	177,14	1,56	59,59
UTC201120280618	7	191,13	-104,94	194,71	133,59	194,2	14,90	1,57	-85,37	1,53	153,07	1,56	36,10
UTC201120280618	8	190,48	-129,58	195,17	108,95	194,45	-10,31	1,57	-110,10	1,54	128,43	1,56	10,89
UTC201120280618	9	190,05	-153,64	195,07	84,80	194,84	-34,95	1,56	-134,16	1,54	104,37	1,55	-13,18
UTC201120280618	0	189,84	-178,85	194,81	60,16	195,51	-59,01	1,56	-158,80	1,54	79,64	1,56	-37,82
UTC201121280618	1	191,24	43,5448	196,25	-77,922	195,2	163,384	1,58	63,5983	1,55	-57,296	1,55	-174,3
UTC201121280618	2	190,18	20,6265	195,24	-100,36	195,13	139,893	1,56	40,107	1,54	-81,36	1,56	161,09
UTC201121280618	3	190,41	-2,2918	194,69	-123,28	195,59	116,975	1,56	17,1887	1,54	-103,8	1,56	138,17
UTC201121280618	4	190,42	-25,783	194,02	-146,2	195,79	94,0563	1,56	-5,7296	1,54	-126,71	1,56	115,26
UTC201121280618	5	191,01	-49,274	194	-170,26	195,53	70,4738	1,56	-29,794	1,54	-150,78	1,57	91,764
UTC201121280618	6	191,38	-73,339	194,24	165,676	194,93	46,9825	1,56	-53,858	1,53	-174,27	1,56	68,182
UTC201121280618	7	191,3	-97,494	194,54	141,612	194,41	22,3454	1,57	-77,922	1,53	161,092	1,56	43,545
UTC201121280618	8	190,82	-120,41	195,11	117,548	194,41	-1,1459	1,57	-100,93	1,54	137,601	1,56	20,054
UTC201121280618	9	190,24	-145,05	195,21	93,4834	194,66	-25,783	1,56	-125,57	1,54	112,964	1,55	-4,584
UTC201121280618	0	191,25	-169,69	196,52	69,9009	196,22	-49,274	1,58	-148,49	1,56	90,6186	1,57	-26,36
UTC201122280618	1	190,54	38,3882	195,43	-83,079	194,81	157,655	1,57	57,8687	1,54	-63,598	1,56	178,85
UTC201122280618	2	190,09	14,3239	195,09	-106,66	195,18	134,163	1,56	34,3775	1,54	-87,09	1,56	155,36
UTC201122280618	3	190,15	-9,1673	194,43	-130,15	195,68	110,099	1,56	10,3132	1,54	-110,67	1,56	131,3
UTC201122280618	4	190,54	-33,805	194,05	-154,22	195,72	86,5166	1,56	-13,751	1,54	-134,74	1,57	107,23
UTC201122280618	5	191,38	-58,442	193,73	-179,43	195,25	61,8794	1,56	-38,961	1,53	-159,37	1,56	83,079
UTC201122280618	6	191,7	-83,652	194,11	155,363	194,58	36,0963	1,57	-64,171	1,53	174,843	1,56	57,296
UTC201122280618	7	191,82	-107,23	194,96	131,299	195,26	12,6051	1,57	-87,663	1,54	151,352	1,56	34,377
UTC201122280618	8	192,32	-130,73	196,52	108,38	195,49	-10,313	1,58	-109,53	1,55	129,58	1,57	12,605
UTC201122280618	9	190,38	-156,51	195,01	81,933	194,97	-37,242	1,56	-137,03	1,54	101,505	1,56	-16,04
UTC201122280618	0	190,15	179,427	194,48	57,8687	195,55	-61,306	1,56	-161,09	1,54	77,9223	1,56	-40,11