UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

GUILHERME SUARDI CALIN

AMPLIAÇÃO DE FUNCIONALIDADES DE UNIDADE DE MEDIÇÃO FASORIAL SINCRONIZADA DE BAIXO CUSTO NO MONITORAMENTO DA REDE ELÉTRICA A NÍVEL DE DISTRIBUIÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

GUILHERME SUARDI CALIN

AMPLIAÇÃO DE FUNCIONALIDADES DE UNIDADE DE MEDIÇÃO FASORIAL SINCRONIZADA DE BAIXO CUSTO NO MONITORAMENTO DA REDE ELÉTRICA A NÍVEL DE DISTRIBUIÇÃO

Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Gustavo Weber Denardin

PATO BRANCO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 13 horas e 50 minutos do dia 06 de dezembro de 2018, na sala V104, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Gustavo Weber Denardin (orientador), Giovanni Alfredo Guarnieri e Fábio Luiz Bertotti para avaliar o trabalho de conclusão de curso com o título **Ampliação de funcionalidades de unidade de medição fasorial sincronizada no monitoramento da rede elétrica a nível de distribuição**, do aluno **Guilherme Suardi Calin**, matrícula 01374702, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Gustavo Weber Denardin
Orientador (UTFPR)

_____
Giovanni Alfredo Guarnieri
(UTFPR)

_____
Fábio Luiz Bertotti
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

CALIN, Guilherme S. AMPLIAÇÃO DE FUNCIONALIDADES DE UNIDADE DE MEDIÇÃO FASORIAL SINCRONIZADA NO MONITORAMENTO DA REDE ELÉTRICA A NÍVEL DE DISTRIBUIÇÃO. 2018. 60f. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2018.

Sendo um grande e complexo sistema dinâmico e interconectado, o sistema elétrico de potência brasileiro apresenta um desafio para garantir critérios de confiabilidade e robustez, e o conceito de *smart grid* mostra-se como uma solução para atender esses critérios. *Smart grid* é um sistema elétrico moderno que possui recursos como comunicação e controle automático avançado, mas antes que a automação da rede elétrica tradicional para *smart grid* seja realizada, é necessário conhecer o estado do sistema de ponta a ponta, atualmente realizado por sistemas SCADA. Entretanto, esses sistemas possuem atrasos na ordem de segundos, o que impossibilita conhecer o estado do sistema em tempo real, motivando o estudo de soluções alternativas de monitoramento em tempo real. Uma das soluções encontradas foi a unidade de medição fasorial sincronizada, que realiza o monitoramento em tempo real do sistema de distribuição a partir da estimação de fasores, sincronizada com pulso de GPS. Os dados coletados são encapsulados em *frames* de dados em conformidade com a norma IEEE C37.118.2 - 2011, e enviados a um concentrador de dados via conexão com a Internet para armazenamento e processamento. Este trabalho tem por objetivo a ampliação das funcionalidades de uma unidade de medição sincrofasorial desenvolvida no Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da UTFPR Pato Branco. Os dados coletados pela unidade de medição desenvolvida no PPGEE são enviados ao concentrador de dados via conexão cabeada com a Internet. Os desenvolvimentos deste trabalho contemplam adicionar a funcionalidade de envio de dados coletados pela unidade de medição para o concentrador de dados via Wi-Fi, além de realizar a migração do código do sistema embarcado utilizado na unidade de medição para outro mais moderno, com mais funcionalidades e maior poder de processamento.

Palavras-chave: LaTeX, unidade de medição sincrofasorial, monitoramento em tempo real, sistemas embarcados, redes de computadores.

ABSTRACT

CALIN, Guilherme S. ENHANCEMENT OF FEATURES OF A SYNCHRONIZED PHASOR MEASUREMENT UNIT FOR DISTRIBUTION ELECTRICAL GRID MONITORING. 2018. 60f. Undergraduate thesis, Computer Engineering Graduation Program - Federal Technological University of Paraná, Pato Branco, 2018.

As a large and complex dynamic and interconnected system, the Brazilian electrical grid presents a challenge to meet reliability and robustness criteria, and the concept of smart grid shows a solution to meet these criteria. Smart grid is a modern electrical grid with resources like communication and advanced automatic control, but before the automation from the traditional electrical grid to smart grid can be made, it is necessary an end-to-end system monitoring, now made by SCADA systems. However, the monitoring through these systems presents delays in the order of seconds, becoming impossible to know the state of the electrical grid in real time, motivating the study of alternative solutions for real-time monitoring. One of the solutions found is the synchronized phasor measurement unit, which performs a real-time monitoring of distribution grids, through phasor estimation, synchronized with GPS pulse. The collected data is encapsulated in data frames, as written in the IEEE C37.118.2 - 2011 norm, and sent to a data concentrator through Internet connection for storage and processing. This work presents an enhancement of features of a synchronized phasor measurement unit developed in the Electrical Engineering Post-Graduate Program (EEPGP) of Federal Technological University of Paraná, Pato Branco. The data collected by the measurement unit developed in the EEPGP is sent to a data concentrator through wired Internet connection. This work aims to add the feature of sending data collected by the measurement unit to the data concentrator through Wi-Fi, as well as migrating the source code of the embedded system used in the measurement unit to a newer one, with more features and better processing power.

Keywords: LaTeX, synchronized phasor measurement unit, real-time monitoring, embedded systems, computer networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E SIGLAS

AP	*Access Point*, ou Ponto de Acesso.
CMSIS	*Cortex Microcontroller Software Interface Standard*, ou Padrão de Interface de *Software* de Microcontroladores Cortex.
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance* ou Acesso Múltiplo por Detecção de Portadora com Prevenção de Colisão.
DFT	*Discrete Fourier Transform*, ou Transformada Discreta de Fourier.
GPS	*Global Positioning System*, ou Sistema de Posicionamento Global.
HTTP	*Hypertext Transfer Protocol*, ou Protocolo de Transferência de Hipertexto.
IDE	*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado.
IP	*Internet Protocol*, ou Protocolo de Internet.
ISM	*Industrial, Scientific and Medical*, ou Industrial, Científico e Médico.
ISO	*International Standards Organization*, ou Organização Internacional de Padrões.
LSB	*Least Significant Bit*, ou *Bit* Menos Significativo.
MSB	*Most Significant Bit*, ou *Bit* Mais Significativo.
OSI	*Open System Interconnection*, ou Interconexão de Sistema Aberto.
PDC	*Phasor Data Concentrator*, ou Concentrador de Dados Fasoriais.
PMU	*Phasor Measurement Unit*, ou Unidade de Medição Fasorial.
PPS	*Pulse per Second*, ou Pulso por Segundo.
RTI	Rotina de Tratamento de Interrupção.
RTOS	*Real Time Operating System*, ou Sistema Operacional de Tempo Real.
SCADA	*Supervisory Control and Data Aquisition*, ou Controle Supervisório e Aquisição de Dados.
SEP	Sistema Elétrico de Potência.
SMFS	Sistema de Medição Fasorial Sincronizada.
SPI	*Serial Peripheral Interface*, ou Interface Serial Periférica.
TCP	*Transport Control Protocol*, ou Protocolo de Controle de Transporte.
UDP	*User Datagram Protocol*, ou Protocolo de Datagrama do Usuário.

LISTA DE SÍMBOLOS

r Comprimento do vetor

ϕ Ângulo de fase

ω Velocidade ou frequência angular

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Sendo um grande e complexo sistema dinâmico e interconectado, o Sistema Elétrico de Potência (SEP) brasileiro apresenta um desafio para os engenheiros de Sistemas de Potência, dada a complexidade de se garantir critérios de robustez e confiabilidade de um sistema de tal porte. Para que esses critérios sejam atendidos, o conceito de *smart grid*, que é uma rede elétrica moderna dotada de recursos como comunicação e controle automático avançado, mostra-se como uma solução. Entretanto, antes que a automação da rede elétrica tradicional seja realizada, é necessário conhecer o sistema de ponta a ponta (LO; ANSARI, 2012).

O atual sistema de monitoramento, composto por Sistemas de Controle Supervisório e Aquisição de Dados (SCADA) possui tempos de atraso na ordem de segundos, impedindo que se saiba o estado do sistema em tempo real. Devido a isso, soluções alternativas passaram a ser buscadas para encontrar um sistema de monitoramento em tempo real, sendo a unidade de medição fasorial sincronizada uma dessas alternativas.

Uma unidade de medição fasorial (PMU), é um sistema cujo objetivo é monitorar as redes de distribuição e transmissão do SEP. Os dados coletados pela PMU são utilizados para avaliar propriedades da rede elétrica, como frequência, amplitude e fase, o que permite identificar como a rede se comporta em relação aos distúrbios e variações de carga. É essa análise que possibilita traçar um perfil de consumo e identificar causas para tais distúrbios, oferecendo subsídios para estudos sobre como melhor dimensionar a rede. Os dados coletados por uma PMU são enviados a um banco de dados, conhecido como Concentrador de Dados Fasoriais (PDC), para armazenamento e processamento das informações recebidas.

O monitoramento se baseia na estimação de fasores, realizada pela unidade. Um fasor é uma ferramenta matemática utilizada para análise de circuitos elétricos em corrente alternada, e constitui-se de maneira simples para analisar circuitos lineares excitados por fontes senoidais. Encontrar soluções para circuitos desse tipo seria impraticável de outra forma (SADIKU; ALEXANDER, 2013).

Dada uma referência, fasor é um número complexo que representa a amplitude e fase de uma senoide, podendo ser analisado tanto em coordenadas retangulares, quanto em coordenadas polares (SADIKU; ALEXANDER, 2013). Portanto, o fasor é uma importante ferramenta, não só pelas informações que contém, mas também

pelos dados que podem ser adquiridos a partir dele, como fluxo de energia em linhas de distribuição e transmissão, limites de tensão, entre outros. Mais detalhes sobre fasor e sua aplicação na unidade de medição sincrofasorial podem ser encontrados no Capítulo 2.

Sincrofasor é o termo adotado para se tratar da tecnologia que utiliza representação fasorial a partir de uma referência temporal única, obtida a partir do sinal de pulso por segundo proveniente de um GPS (IEEE, 2011b). Dessa forma, é possível sincronizar medidas de várias PMUs em localizações geográficas distintas, todas sob uma mesma referência. Apesar das PMUs serem uma alternativa ao monitoramento realizado por sistemas SCADA, as unidades comerciais possuem elevado custo na ordem de milhares de dólares, com o agravante de que são necessárias várias unidades para um monitoramento abrangente. Isso dificulta a popularização de seu uso.

Motivado pelo elevado custo das PMUs comerciais, a dissertação de mestrado de Flávio Lori Grando(GRANDO, 2016) apresentou a proposta de uma arquitetura de baixo custo para desenvolvimento de uma unidade de medição fasorial sincronizada. Essa unidade foi desenvolvida em conformidade com todos os testes previstos pela norma IEEE C37.118.1 - 2011, utilizando o *kit* de desenvolvimento STM32F429I-*Discovery*. A coleta de dados é realizada ao ligar a unidade em uma tomada trifásica, e os resultados são enviados a um PDC por meio de conexão cabeada com a Internet por meio de um módulo *Ethernet* externo, pois o *kit* utilizado não possui módulo *Ethernet* próprio.

Entretanto, a utilização de um módulo externo reduz confiabilidade e robustez no envio de dados a um PDC, e a ocorrência de falhas no envio significa perda permanente desses dados, pois o *kit* STM32F429I-*Discovery* não possui memória suficiente para armazenamento temporário até que os dados sejam de fato recebidos pelo PDC. A recepção desses dados é de suma importância para que possam ser analisados.

Dada a importância do envio de dados a um PDC, a PMU de Grando (2016) apresenta uma limitação em sua flexibilidade de utilização, pois além de ser necessário que a unidade esteja ligada a uma tomada trifásica, o único meio de envio de dados a um PDC é por conexão cabeada com a Internet.

Levando em consideração os fatores apresentados, este trabalho visa melhorar e ampliar as funcionalidades da PMU desenvolvida por Grando (2016). Visto que o *kit* STM32F429I-*Discovery* não possui um módulo *Ethernet* integrado, será realizada a migração do código para o *kit* STM32F746G-*Discovery*. Além de possuir todas as funcionalidades necessárias para o funcionamento da PMU, esse *kit* também

possui um módulo *Ethernet* próprio, bem como outras funcionalidades não existentes no STM32F429I-*Discovery*, como tela gráfica sensível ao toque de 4,3 polegadas e suporte a cartão SD, criando assim possibilidades para trabalhos futuros.

Entretanto, a migração do código para outro *kit* melhora a confiabilidade e robustez no envio de dados a um PDC por conexão cabeada com a Internet, mas não aumenta a flexibilidade de utilização da PMU em si. Para isso, será implementada a funcionalidade de envio de dados a um PDC também por conexão sem fio à Internet, por Wi-Fi, utilizando-se o *kit* ESP32WROOM. Dessa forma, a PMU, com o *kit* STM32F746G-Discovery, poderá enviar os dados a um PDC por conexão cabeada com a Internet de maneira mais confiável, ou enviar os dados ao *kit* ESP32WROOM, para que esse efetue o envio de dados ao PDC por Wi-Fi.

1.2 OBJETIVO GERAL

Ampliar as funcionalidades da unidade de medição já desenvolvida por Grando (2016), migrando o código para outro *kit* com mais recursos e dotado de módulo Ethernet próprio, além de adicionar a funcionalidade de envio de dados a um PDC por Wi-Fi.

1.2.1 OBJETIVOS ESPECÍFICOS

Os objetivos deste trabalho, e o detalhamento de suas etapas, são mostrados nos tópicos a seguir:

- Migrar o código da PMU desenvolvida por Grando (2016) com o *kit* STM32F429I-*Discovery* para o kit STM32F746G-*Discovery*, que possui mais recursos, maior poder de processamento e um módulo *Ethernet* próprio;
- Realizar o estudo de comunicação SPI dos *kits* STM32F746G-*Discovery* e ESP32WROOM;
- Implementar a comunicação SPI entre os *kits* STM32F746G-*Discovery* e ESP32WROOM, de forma que os dados coletados pela PMU possam ser enviados ao ESP32WROOM;
- Implementar a funcionalidade de envio dos dados recebidos pela PMU a um PDC por Wi-Fi;

1.3 ESTRUTURA DO TRABALHO

O documento está estruturado em 6 capítulos. O Capítulo 2 se destina a parte de revisão bibliográfica e pesquisa teórica quanto aos assuntos pertinentes a este trabalho. O Capítulo 3 se destina aos materiais e métodos utilizados, bem como apresentação ilustrativa do sistema proposto. O Capítulo 4 apresenta como foi realizado o desenvolvimento do trabalho, o Capítulo 5 segue para os resultados alcançados e o Capítulo 6 encerra este trabalho comparando o que foi proposto e o que foi de fato alcançado.

2 REFERENCIAL TEÓRICO

Neste capítulo, será realizada a fundamentação teórica dos conceitos necessários à contextualização e realização deste trabalho. A seção 2.1 apresenta o conceito de fasor e seu detalhamento matemático. A seção 2.2 explica com mais detalhes uma PMU, justificando seu surgimento e seu funcionamento, e posteriormente apresenta a arquitetura proposta por Grando (2016) e as oportunidades de ampliação de funcionalidades que este trabalho visa realizar.

A seção 2.3 explica o funcionamento da interface de comunicação escolhida para a troca de informações entre os dois sistemas embarcados presentes neste trabalho: a comunicação SPI. As seções 2.4, 2.5 e 2.6 se destinam aos conceitos necessários ao envio de dados da PMU a um PDC por meio de conexão com a Internet, primeiro explicando o conceito de arquitetura de redes baseado em modelo de camadas, bem como a função de cada uma delas (Seção 2.4), posteriormente explicando o protocolo da camada de enlace a ser utilizado neste trabalho: o Wi-Fi (Seção 2.5), e por fim, será realizada uma breve análise da interface entre a camada de aplicação e transporte na seção 2.6 (*sockets*).

2.1 FASOR

Um fasor é uma representação matemática para tornar mais simples as operações matemáticas com sinais senoidais. Segundo Sadiku (2013), ”Fasor é um número complexo que representa a amplitude e a fase de uma senoide”. Também segundo Sadiku (2013), ”Os fasores se constituem de maneira simples para analisar circuitos lineares excitados por fontes senoidais; encontrar a solução para circuitos desse tipo seria impraticável de outro modo”.

Dado que um fasor é um número complexo, é necessário revisar algumas propriedades e equações (SADIKU; ALEXANDER, 2013). Um número complexo *z* na forma retangular pode ser escrito como

$$z = x + jy \text{ ,} \tag{1}$$

sendo *x* a parte real do número, e *y* a parte imaginária. Um número complexo *z* também pode ser representado nas formas polar ou exponencial, respectivamente:

$$z = r\angle\phi \; ; \tag{2}$$

$$z = re^{j\phi} \; . \tag{3}$$

A relação entre a forma retangular e forma polar é apresentada na Figura 1.

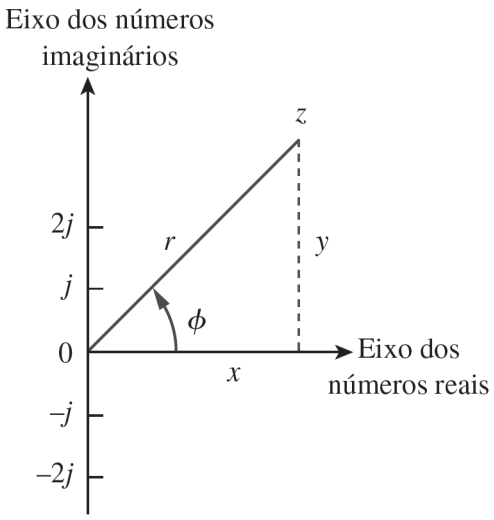

Figura 1: Eixo dos números complexos

Fonte: Sadiku e Alexander (2013)

Analisando o triângulo retângulo apresentado na Figura 1 e utilizando-se de trigonometria e teorema de Pitágoras, tem-se que r e ϕ podem ser expressos como

$$r = \sqrt{x^2 + y^2} \; ; \tag{4}$$

$$\phi = \arctan\left(\frac{y}{x}\right) \; . \tag{5}$$

Da mesma forma que, conhecendo os valores de r e ϕ, é possível encontrar os valores de *x* e *y* por meio de

$$x = rcos(\phi) \; ; \tag{6}$$

$$y = rsen(\phi) \; . \tag{7}$$

Portanto, ao substituir (6) e (7) em (1), tem-se que

$$z = x + jy = r(cos(\phi) + jsen(\phi)) \; . \tag{8}$$

Segundo Sadiku (2013), "A adição e subtração de números complexos são mais bem realizadas na forma retangular; a multiplicação e a divisão são mais bem efetuadas na forma polar.". Adição e subtração são operações mais imediatas, pois basta somar ou subtrair os termos reais entre si, e de forma análoga com os termos imaginários. Já as outras operações mais comuns são listadas a seguir, tomando dois números complexos z_1 e z_2 como exemplos:

Tabela 1: Operações com números complexos

Multiplicação:	$z_1 z_2 = r_1 r_2 \underline{/\phi_1 + \phi_2}$
Divisão:	$\frac{z_1}{z_2} = \frac{r_1}{r_2} \underline{/\phi_1 - \phi_2}$
Inversão:	$\frac{1}{z_1} = \frac{1}{r_1} \underline{/-\phi_1}$
Raiz quadrada:	$\sqrt{z_1} = \sqrt{r_1} \underline{/\frac{\phi}{2}}$
Conjugado Complexo:	$z_1^* = x_1 - jy_1 = r_1 \underline{/-\phi_1} = r_1 e^{-j\phi_1}$

A representação de fasor se baseia na identidade de Euler, que é

$$e^{\pm j\phi} = cos(\phi) \pm jsen(\phi) \ . \tag{9}$$

A identidade de Euler, representada em (9), pode ser separada nas partes real e imaginária, respectivamente:

$$Re\left(e^{\pm j\phi}\right) = cos(\phi) \ ; \tag{10}$$

$$Im\left(e^{\pm j\phi}\right) = \pm sen(\phi) \ . \tag{11}$$

Agora, suponha que uma tensão $v(t)$ é dada por uma cossenoide genérica, cuja equação é dada por

$$v(t) = V_M cos(\omega t + \phi) \ , \tag{12}$$

em que V_M é a amplitude da cossenoide. Como apresentado em (10), pode-se representar (12) como sendo

$$V_M cos(\omega t + \phi) = Re\left(V_M e^{j\phi} e^{j\omega t}\right) \ , \tag{13}$$

em que

$$\boldsymbol{V} = V_M e^{j\phi} = V_M \underline{/\phi} \ . \tag{14}$$

Portanto, (12) pode ser representada como

$$v(t) = Re\left(\boldsymbol{V}e^{j\omega t}e^{j\phi}\right) \ . \tag{15}$$

Em que $\boldsymbol{V}$ é a representação fasorial da cossenoide $v(t)$. Segundo Sadiku (2013), "Toda vez que uma senoide for expressa como fasor, o termo $e^{j\omega t}$ está implicitamente presente". Para desenvolver o fasor, tanto (10) quanto (11) podem ser usadas, mas é mais comum utilizar-se da parte real da identidade de Euler ao invés da parte imaginária. Portanto, é mais intuitivo trabalhar com cossenos ao invés de senos. Visto que a diferença entre um seno e cosseno é apenas a fase, é possível transformar um seno em cosseno, e vice-versa, por meio das seguintes relações, retiradas do livro de Sadiku (2013):

$$\begin{aligned} sen(\omega t \pm 180^{\circ}) &= -sen(\omega t) \ ; \\ cos(\omega t \pm 180^{\circ}) &= -cos(\omega t) \ ; \\ sen(\omega t \pm 90^{\circ}) &= \pm cos(\omega t) \ ; \\ cos(\omega t \pm 90^{\circ}) &= \mp sen(\omega t) \ . \end{aligned} \tag{16}$$

Para analisar melhor a representação fasorial de (14) e (15), observe a Figura 2.

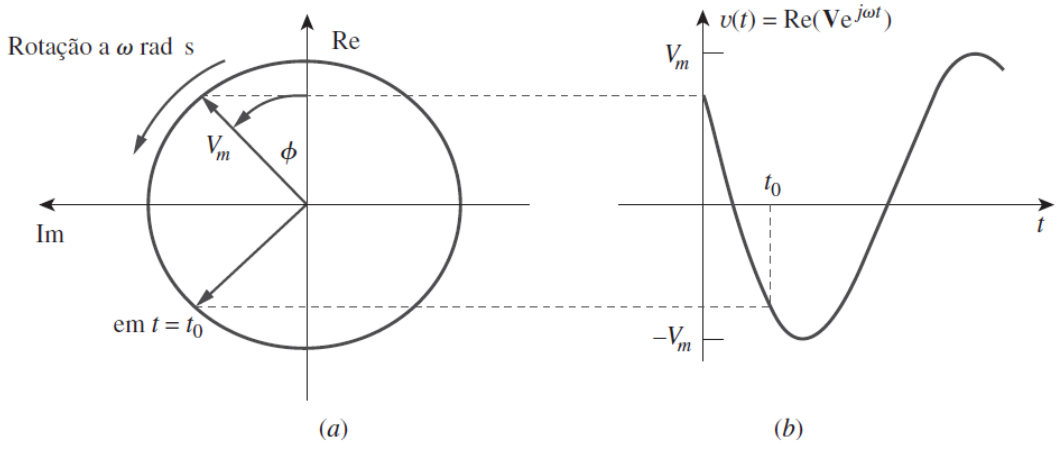

Figura 2: Representação de $Ve^{j\omega t}$. a) fasor girando no sentido anti-horário. b) sua projeção no eixo real, em função do tempo

Fonte: Sadiku e Alexander (2013)

A medida que o tempo cresce, o fasor da Figura 2(a) gira em um círculo de raio V_m, em uma velocidade angular ω no sentido anti-horário. Já na Figura 2(b), é apresentada sua projeção no eixo real em função do tempo (SADIKU; ALEXANDER, 2013). O termo $e^{j\omega t}$, que dita sua frequência angular, fica implícito ao se representar um seno ou cosseno como fasor, e é suprimido, pois ω é constante. Porém, a resposta

depende dessa frequência angular, logo, o domínio dos fasores também é conhecido como domínio da frequência (SADIKU; ALEXANDER, 2013). Por tais motivos, é de suma importância ter em mente a frequência angular ω ao se trabalhar com fasores.

Basicamente, para obter o fasor correspondente a uma senoide, transforma-se o seno em cosseno, e utiliza-se a informação de magnitude e fase, suprimindo o termo $e^{j\omega t}$. (13) é uma representação no domínio do tempo, enquanto (14) é a representação no domínio dos fasores.

Como um fasor possui magnitude e fase, ele se comporta como vetor e é convencionalmente escrito em negrito (SADIKU; ALEXANDER, 2013). Uma representação gráfica de fasores é conhecida como diagrama fasorial, exemplificado na Figura 3 supondo dois fasores genéricos $\boldsymbol{V} = V_m\underline{/\phi}$ e $\boldsymbol{I} = I_m\underline{/-\phi}$.

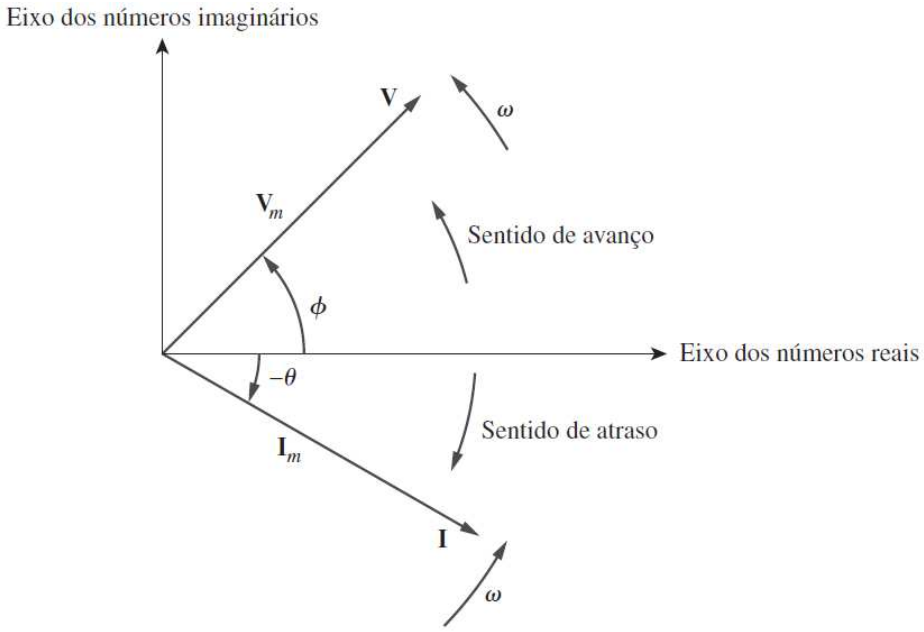

Figura 3: Diagrama fasorial apresentando $V = V_m\underline{/\phi}$ e $I = I_m\underline{/-\phi}$

Fonte: Sadiku e Alexander (2013)

Por fim, deve-se ter em mente que a análise por fasores se aplica apenas quando a frequência é constante. Além disso, a manipulação de dois ou mais sinais senoidais, quando representados por fasores, é possível, mas é necessário que suas frequências sejam as mesmas. Dessa forma, fasor é uma ferramenta muito importante em análise de circuitos em corrente alternada. Visto que o domínio dos fasores também é o domínio da frequência para uma frequência constante, é possível transformar elementos passivos como resistores, capacitores e indutores em impedâncias no domínio da frequência, e realizar toda a análise de uma maneira mais simples. Como

dito por Sadiku (2013), "A ideia de expressar informações em domínios alternados é fundamental em todas as áreas da engenharia.".

2.2 UNIDADE DE MEDIÇÃO FASORIAL SINCRONIZADA

Uma PMU é um sistema que realiza o monitoramento da rede elétrica a nível de distribuição e transmissão. A necessidade de tal monitoramento tem sua origem a partir da rede elétrica tradicional. O SEP brasileiro consiste de um sistema complexo, com um conjunto de elementos interconectados que provocam uma interação dinâmica entre si.

Diante de um sistema de tamanha complexidade, surgiu o conceito de *smart grids*, que pode ser definido como a transformação da rede elétrica tradicional em uma rede elétrica modernizada que integra comunicação, controle automático avançado, inteligência artificial, processamento de sinais e tecnologia da informação (BHUIYAN *et al.*, 2017). Isso torna necessário um monitoramento eficaz de ponta a ponta de todo o sistema, seja para tomada de decisões ou para estudos de melhor dimensionamento.

Essa necessidade de monitoramento em tempo real é o que motiva estudos de soluções alternativas ao atual modelo de monitoramento do SEP, composto por sistemas SCADA. Os dados são obtidos por telemedição em intervalos regulares de tempo, porém não sincronizados. Além disso, os tempos de atraso desse tipo de sistema são na ordem de segundos, o que impossibilita conhecer o estado do sistema em tempo real (BERG *et al.*, 2015).

Uma das soluções alternativas é a medição fasorial sincronizada, que se baseia na estimação de fasores. Os ângulos dos fasores de tensão nos barramentos do sistema sempre foram de especial interesse para os engenheiros de sistemas de potência, pois é bem conhecido que o fluxo de energia em uma linha está relacionado à diferença de fase das tensões nos terminais da linha (GRAINGER; STEVENSON, 1994). Assim, a partir das tensões complexas, é possível calcular todas as grandezas necessárias para avaliar carregamento de linhas, limites de tensão e geração, intercâmbio entre áreas, etc.

Mas para que essa medição seja sincronizada, é necessária uma referência temporal em comum para todos os pontos do sistema, que possui proporções geográficas imensas já que o Brasil é um dos maiores países do planeta. Além disso, é necessário que essa referência seja precisa. Isso é possível graças ao Sistema de

Posicionamento Global (GPS). Ele possui referência temporal com precisão obtida por meio do relógio atômico dos satélites, que fornecem o sinal de pulso por segundo (PPS) aos receptores na Terra, possibilitando o sincronismo das medidas.

Portanto, um Sistema de Medição Fasorial Sincronizada (SMFS) é composto por várias PMUs espalhadas pela rede, para realizarem a medição dos fasores de tensão e corrente sob uma mesma referência temporal. Os dados coletados são enviados a PDCs para armazenamento e posterior processamento, conforme exemplificado na Figura 4.

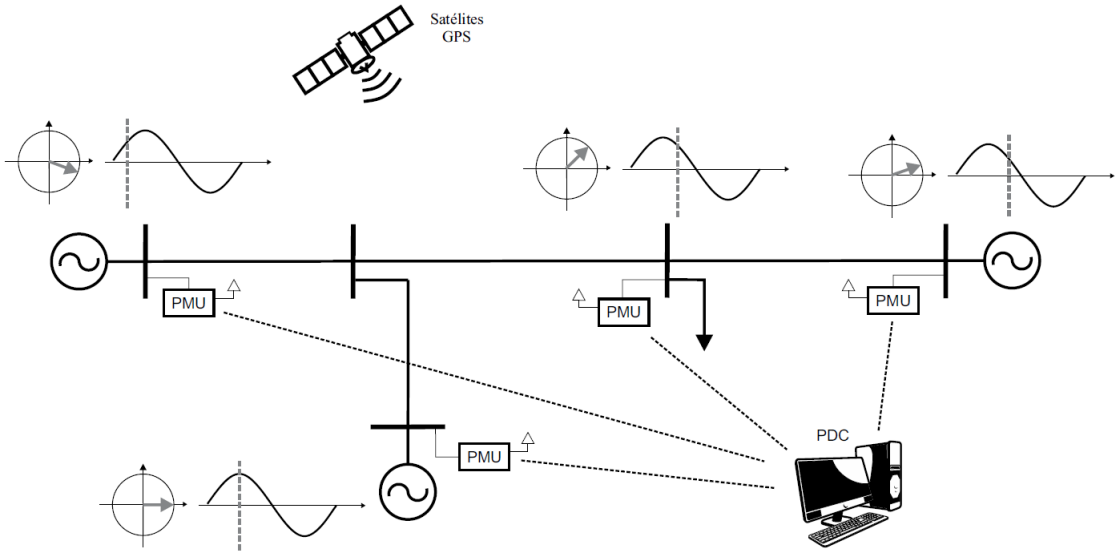

Figura 4: Medição fasorial sincronizada do sistema elétrico

Fonte: Grando (2016)

A Figura 5 ilustra a diferença de monitoramento entre os dados de sistemas SCADA e os dados de uma PMU, durante um distúrbio na rede elétrica norte-americana (BERG *et al.*, 2015).

Ao analisar o gráfico da Figura 5, é notável como os resultados obtidos pela PMU captam o comportamento dinâmico da rede elétrica com maior resolução do que os dados obtidos por sistemas SCADA.

Todos os dados coletados por uma PMU se baseiam na estimação de fasores. Esses dados são coletados por meio de um sistema embarcado, em que são necessárias várias amostras do sinal elétrico e aplicação de processos matemáticos. Isso torna necessário a utilização de algum algoritmo para realizar essa estimação fasorial. A técnica escolhida por Grando (2016) na PMU proposta é a Transformada Discreta de Fourier (DFT).

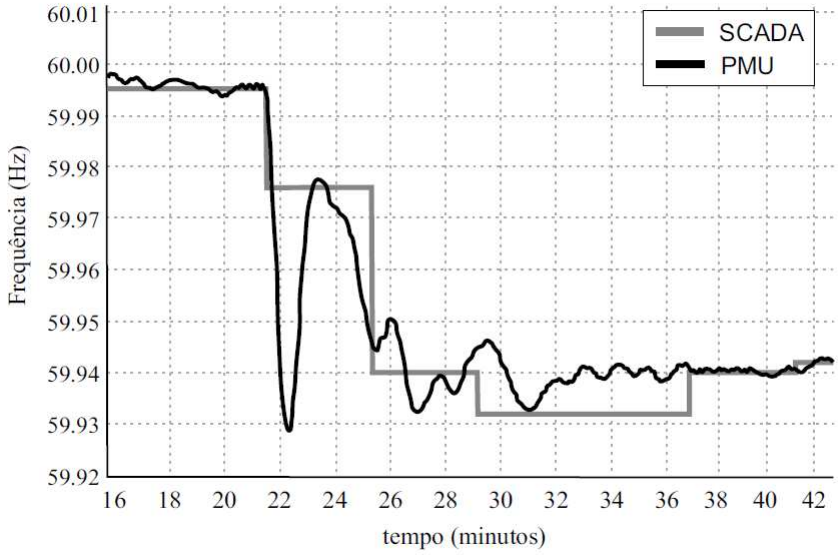

Figura 5: Comparativo entre a medição de um sistema SCADA e a de uma PMU

Fonte: Adaptado de Berg *et al.* (2015)

Com a utilização da DFT, é possível extrair o fasor da componente fundamental do sinal, porém, esse método prejudica o resultado da estimação fasorial quando a frequência fundamental é diferente da frequência de operação do sistema (GRANDO, 2016). Portanto, é necessária alguma otimização para melhorar o desempenho da estimação sob condições dinâmicas. A estratégia de otimização do cálculo da DFT utilizada na PMU proposta por Grando (2016) é baseada em *hardware*, em que a taxa de amostragem varia de maneira sincronizada com a frequência do sinal.

Visto que não é possível extrair a informação de frequência de uma senoide por meio de fasores, há também uma outra etapa específica para estimação de frequência. Para tal, é utilizado um cálculo dado pela variação de fase entre duas estimações fasoriais adjacentes (IEEE, 2011b).

Todo o detalhamento matemático da estimação de fasores e estimação de frequência de um sinal elétrico, na arquitetura proposta por Grando (2016), pode ser encontrado no capítulo 2 de seu documento de dissertação de mestrado, e mais detalhes sobre otimização em *hardware* da DFT podem ser encontrados na seção 4.4.1.

A estrutura genérica de uma PMU é basicamente composta de filtros *anti-aliasing*, módulo de conversão A/D para aquisição de amostras, receptor GPS para o sincronismo das amostras coletadas, um microprocessador para realizar a estimação de fasores e frequência por meio de algoritmos a partir de tais amostras, e um transceptor de comunicação para enviar os dados coletados pela PMU a um PDC. Essa estrutura genérica é apresentada na Figura 6.

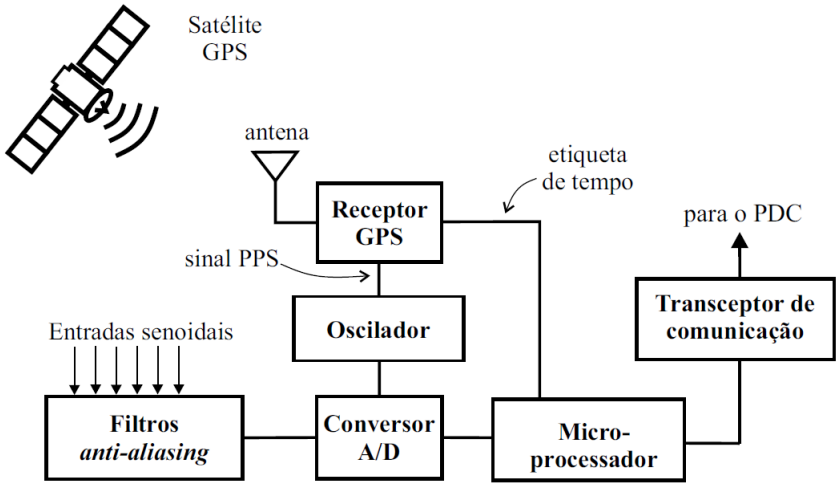

Figura 6: Estrutura genérica de uma PMU

Fonte: Adaptado de Phadke e Thorp (2008)

O receptor GPS recebe o sinal do satélite, sendo responsável por acionar os conversores A/D e fornecer o sincronismo das medidas. Como o sinal PPS é gerado uma vez por segundo, é possível o uso de um oscilador para manter a precisão temporal na ausência de satélites visíveis, ou para gerar pulsos na frequência desejada.

Após a aquisição das amostras, elas são processadas por meio de algoritmos específicos responsáveis por estimar módulo, fase e frequência de tensão e corrente. Nesse caso, o módulo GPS também oferece a etiqueta de tempo para o microprocessador, além do pulso de sincronismo, para que cada fasor estimado possua data e hora com precisão de relógios atômicos dos satélites. Após processados, os dados são empacotados em *frames*, cujo formato é determinado pela norma IEEE C37.118.2 - 2011, e enviados a um PDC através de um canal de comunicação. É possível comparar a estrutura genérica de uma PMU e a PMU proposta por Grando (2016) a partir das Figuras 6 e 7, respectivamente.

PMUs tradicionais medem o sinal dos Transformadores de Corrente e Transformadores de Potencial das subestações. Transformador de Corrente reproduz, no secundário, a corrente que circula no enrolamento primário em uma proporção reduzida que é definida e adequada, sem alterar sua posição vetorial, para que sinais de alta corrente possam ser medidos em equipamentos de medição de uma maneira mais segura e com menor custo. Transformadores de Potencial operam de maneira análoga, mas para medir níveis de tensão ao invés de corrente. Podem ser usados

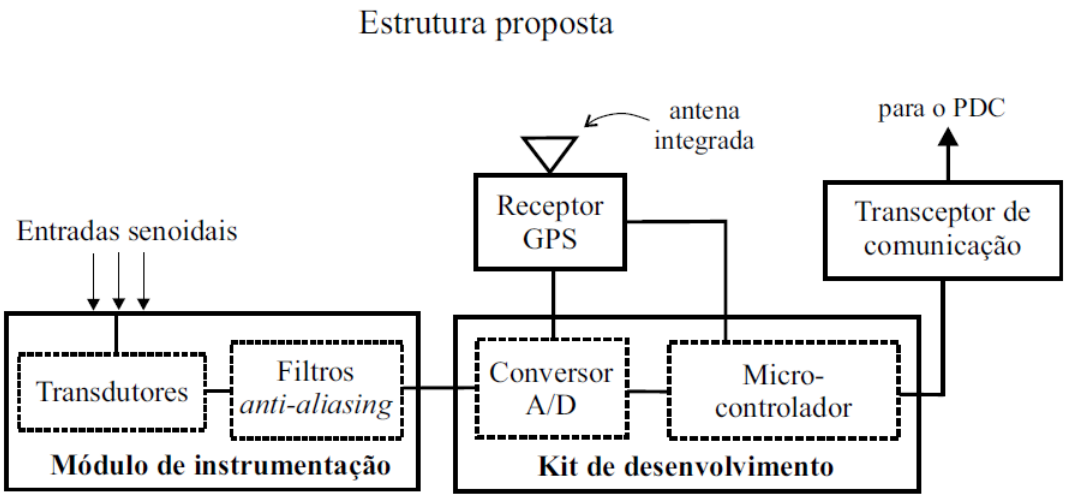

Figura 7: Estrutura proposta

Fonte: Grando (2016)

tanto para níveis de baixa tensão comuns à área de Engenharia Elétrica, como 0,6kV, quanto para alta tensão como 24,2kV.

A PMU proposta por Grando (2016) objetiva extrair o sinal de baixa tensão da rede elétrica ($127V_{RMS}$ e 60Hz) por meio de tomadas trifásicas, sendo utilizados transdutores para reduzir a amplitude do sinal a níveis suportados pelos conversores A/D do sistema embarcado utilizado. Juntamente com um filtro *anti-aliasing*, tem-se o módulo de instrumentação.

É interessante ressaltar que foram utilizados três conversores A/D separados, e não apenas canais diferentes, para que seja possível a aquisição de dados referentes as três fases do sistema elétrico. Um dos conversores é configurado como mestre, e os outros dois são configurados como escravos, para controlar a aquisição de amostras. As Figuras 8 e 9 ilustram o processo de aquisição na arquitetura proposta por Grando (2016).

A Figura 8 apresenta um esquema simplificado de como os dados são coletados, e o que exatamente é realizado pelas etapas de *hardware* e *software*. Após as tensões da rede serem condicionadas no Módulo de Instrumentação, os sinais são adquiridos conforme Figura 9. O receptor GPS, que possui oscilador próprio e antena integrada, é responsável por receber o sinal PPS para acionar os conversores A/D e dar início ao intervalo de aquisição, de responsabilidade da etapa de *hardware*. Após a aquisição das amostras, inicia-se o intervalo de processamento, de responsabilidade da etapa de *software*, para estimação de fasores e frequência. Por fim, o transceptor

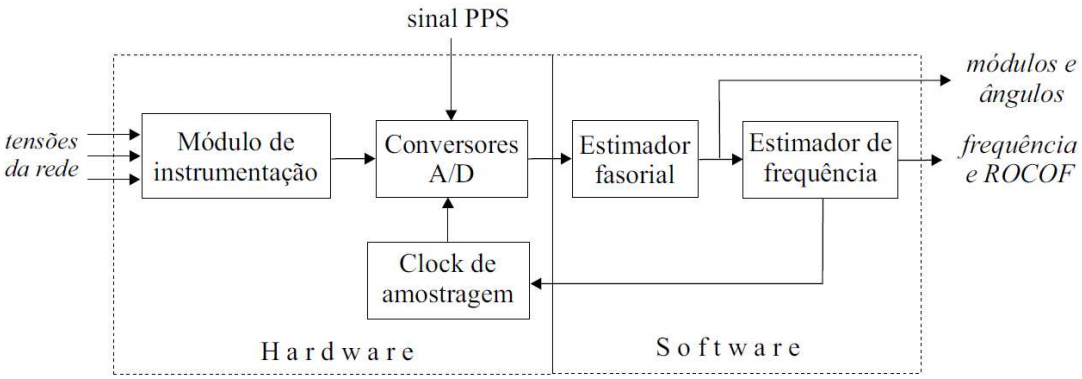

Figura 8: Esquema simplificado para estimação dos dados pertinentes

Fonte: Grando (2016)

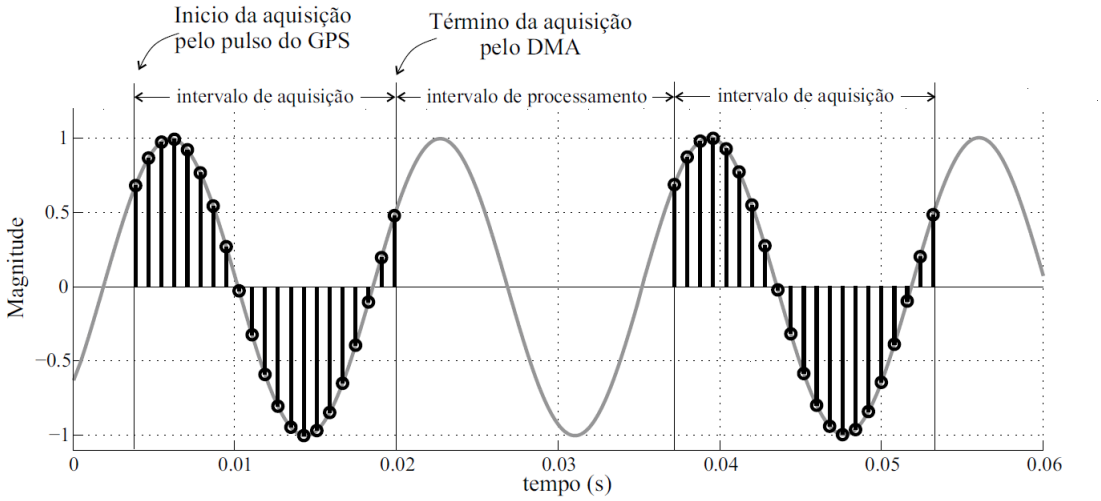

Figura 9: Estratégia de amostragem

Fonte: Grando (2016)

de comunicação, correspondente ao módulo *Ethernet* externo utilizado, é responsável por enviar os dados coletados a um PDC, via conexão cabeada com a Internet.

2.3 COMUNICAÇÃO SPI

A Interface Serial Periférica (SPI) é uma especificação de comunicação síncrona, serial e bidirecional entre um dispositivo mestre e um ou mais dispositivos escravos (NOVIELLO, 2016). Foi definido pela Motorola, e é a interface de comunicação síncrona mais comum em uso geral, utilizado em uma vasta gama de aplicações (DAVIES, 2008). O problema é que não possui um protocolo fixo, permitindo variações de configuração entre um dispositivo e outro. Dessa forma, é necessário sempre con-

sultar o *data sheet* de um dispositivo que use SPI, para compreender o protocolo utilizado.

SPI possui duas linhas para troca de dados e uma linha para sinal de *clock*, sendo também possível um quarto sinal para seleção de dispositivo escravo (PEREIRA, 2005). Permite a comunicação *full-duplex*, o que significa ser possível transmitir simultaneamente nas duas direções entre dois dispositivos (DAVIES, 2008). É importante ressaltar que, por não possuir um protocolo fixo, é possível que a nomenclatura das linhas mude entre um fabricante e outro. A estrutura de um barramento SPI típico é apresentada na Figura 10.

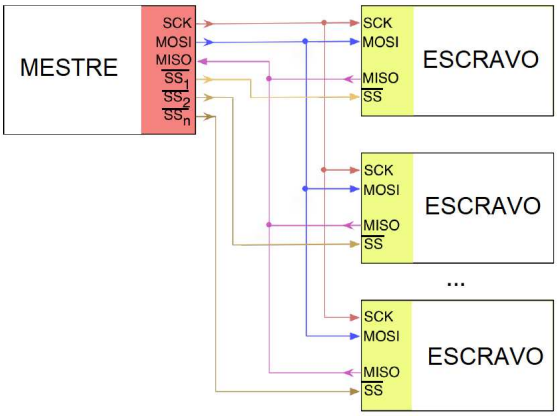

Figura 10: Estrutura de um barramento SPI típico

Fonte: Adaptado de Noviello (2016)

A Figura 10 apresenta um exemplo da ligação entre um dispositivo mestre e três dispositivos escravos. Os sinais utilizados são divididos em:

- **SCK**: É o sinal de *clock* responsável pelo sincronismo da comunicação. **SCK** vem de *Signal Clock* (Sinal de Clock). Outras nomenclaturas incluem **SCLK** ou até mesmo **SPSCK** (*SPI Serial Clock*, ou Clock Serial SPI). Esse sinal é fornecido pelo dispositivo mestre, o que significa que toda transmissão é iniciada por ele.

- **MOSI**: Vem de *Master Output Slave In* (ou Saída do Mestre, Entrada do Escravo). Outras nomenclaturas incluem **SDO** (*Serial Data Out*, ou Saída de Dados Serial), **SO** (*Serial Out*, ou Saída Serial) ou **DOUT** (*Data Out*, ou Saída de Dados). É utilizado para enviar dados do mestre para o escravo.

- **MISO**: Vem de *Master Input Slave Output* (ou Entrada do Mestre, Saída do Escravo). Outras nomenclaturas incluem **SDI** (*Serial Data In*, ou Entrada de Dados Serial), **SI** (*Serial In*, ou Entrada Serial) ou **DIN** (*Data In*, ou Entrada de Dados). É utilizado para enviar dados do escravo para o mestre.

- $\overline{\mathbf{SS_n}}$: A nomenclatura vem de *Slave Select* (Seleção de Escravo), sendo "n"o número de dispositivos escravos conectados ao dispositivo mestre. Outras nomenclaturas incluem **CS** (*Chip Select*, ou Seleção de Chip) ou **CE** (*Chip Enable*, ou Ativação de Chip). É o sinal responsável por selecionar com qual escravo o mestre vai se comunicar naquele instante. Para cada dispositivo escravo conectado, é necessário um pino a mais no dispositivo mestre, para que seja possível escolher com qual deles será realizada a troca de dados. A barra em cima da sigla indica que o sinal é ativado em nível lógico baixo. Porém, podem existir variações de SPI em que o SS é ativado em nível lógico alto.

O funcionamento da comunicação SPI consiste, basicamente, em dois registradores de deslocamento, um em cada dispositivo. O dispositivo mestre fornece o sinal de *clock* para sincronismo, tanto para si quanto para os dispositivos escravos, assim como o sinal de seleção do dispositivo escravo (DAVIES, 2008). A Figura 11 ilustra esse conceito.

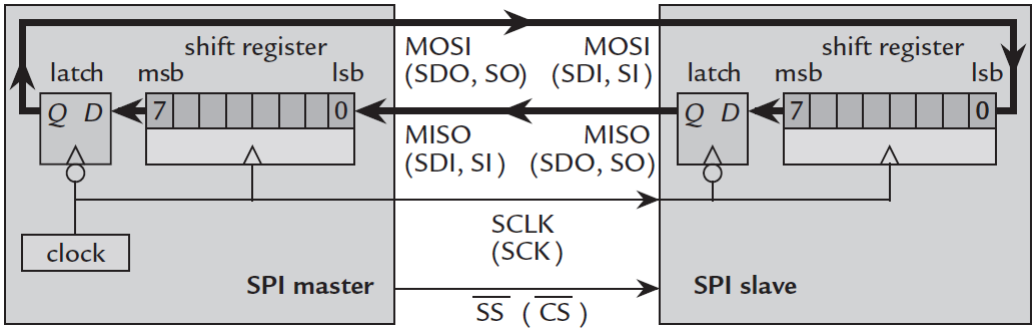

Figura 11: Comunicação SPI entre mestre e escravo. Conceitualmente um registrador de deslocamento

Fonte: Davies (2008)

Um *byte* é o comprimento mais comum para uma transferência, mas também é possível transmitir outras quantidades de dados, como palavras ou *nibbles* (4 *bits*). A Figura 11 apresenta um exemplo de transferência de um *byte*.

Transmissão e leitura de dados do registrador de deslocamento em uma comunicação SPI são realizadas nas bordas de subida ou descida do *clock*. Qual borda realiza qual ação depende do protocolo adotado. Na Figura 11, cada dispositivo lê sua entrada a partir do Bit Menos Significativo (LSB) em uma determinada borda (subida, nesse exemplo), e insere um novo *bit* em sua saída a partir do Bit Mais Significativo (MSB) na borda oposta (descida, nesse caso). Dessa maneira, um *bit* é transferido em cada direção a cada ciclo de *clock*. Depois de oito ciclos, o *byte* é transmitido e

a transferência está completa. É importante ressaltar que, a princípio, transmissão e recepção são inseparáveis, e é necessário transferir um *bit* para poder receber um *bit*.

Devido ao SPI não possuir um protocolo fixo, é necessário ressaltar outro detalhe importante: apesar do exemplo da Figura 11 mostrar a transferência a partir do MSB, podem existir casos em que ocorre o contrário, ou seja, transfere-se o LSB primeiro.

Existem dois outros aspectos importantes a serem acordados em uma comunicação SPI, que são Polaridade de *Clock* e Fase de *Clock*. O mestre e o escravo precisam estar de acordo com a mesma configuração para que a comunicação seja efetuada com sucesso. A Polaridade de *Clock* define qual o estado inativo do *clock*, ou seja, em qual nível lógico ele vai permanecer quando não estiverem sendo realizadas transmissões de dados. Já a Fase de *Clock* define em qual borda será realizada a transmissão (ou escrita), e em qual borda será realizada a leitura.

A Figura 12 demonstra um exemplo de transmissão de um *nibble*, em que a Polaridade de *Clock* define que o estado inativo do *clock* é 0, e a Fase de *Clock* define que leituras são realizadas nas bordas de subida, enquanto as transmissões são realizadas nas bordas de descida.

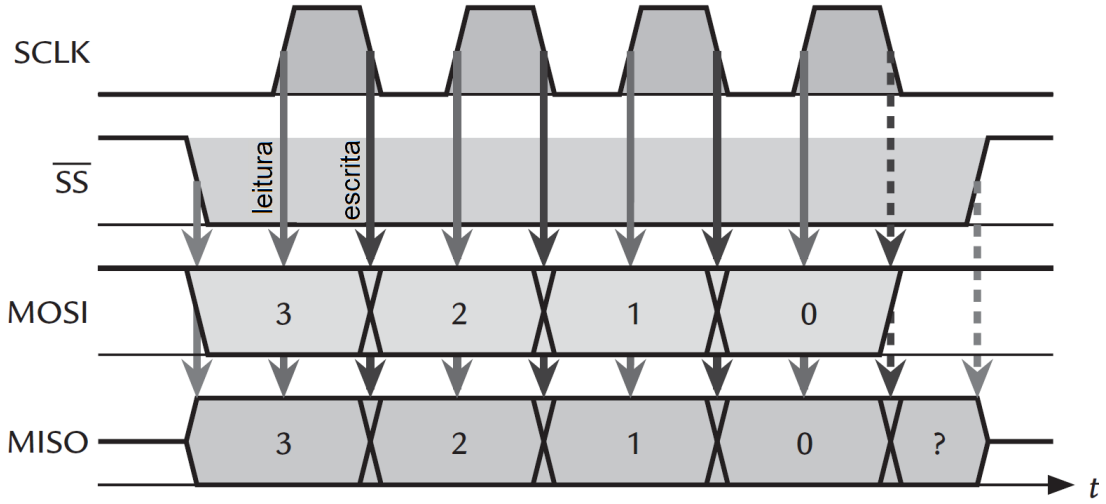

Figura 12: Transferência de um *nibble* para uma configuração específica de polaridade e fase de *clock*

Fonte: Adaptado de Davies (2008)

Mesmo com uma certa quantidade de fatores a serem levados em conta, a comunicação SPI é simples e sua velocidade de transmissão pode chegar a dezenas de *megabits* por segundo, o que o torna ideal quando grandes volumes de dados têm de ser transferidos (DAVIES, 2008), como é o caso deste trabalho.

2.4 INTERNET E ARQUITETURA DE CAMADAS

A Internet é uma rede de computadores que interconecta muitos dispositivos ao redor do planeta, também sendo uma infraestrutura que fornece serviços a aplicações. Essas aplicações incluem serviços de e-mail, navegação na Web, redes sociais, jogos *on-line*, entre outros. (KUROSE; ROSS, 2013)

Com tantos dispositivos possuindo diferentes *hardwares* e sistemas operacionais, e tantas aplicações heterogêneas conectadas entre si, é notável a complexidade de tamanha infraestrutura. Devido a isso, tornou-se necessário encontrar uma maneira de organizar a arquitetura de rede, e para esse fim, surgiram modelos de referência baseado em arquitetura de camadas. Existem três conceitos importantes a serem definidos antes do detalhamento do que é uma arquitetura de camadas e quais são os modelos de referência. São eles: serviço, interface e protocolo (TANENBAUM; WETHERALL, 2011):

- Serviço informa o que a camada faz, mas não como ela funciona e nem como as entidades acima dela a acessam.
- Interface de uma camada dita como os processos acima dela podem acessá-la, além dos parâmetros e resultados esperados, mas não revela o funcionamento interno da camada.
- Protocolo é um conjunto de regras, que define o formato e significado das mensagens trocadas entre as entidades contidas em uma camada. Protocolos utilizados em cada camada são de responsabilidade das mesmas.

A arquitetura de camadas permite organizar um sistema complexo, estruturando-o de tal forma que seja possível deixar bem definidos os serviços de cada camada. Além disso, essa estruturação permite modificar a execução do serviço de uma camada, sem que isso afete as outras. Ou seja, a implementação pode mudar, contanto que uma camada forneça o mesmo serviço para a que está acima dela, e use os mesmos serviços da que está abaixo dela (KUROSE; ROSS, 2013).

A Organização Internacional de Padrões (ISO) definiu o modelo de referência *Open System Interconnection* (OSI) no final da década de 1970. Mesmo antigo, ele ainda é citado pois o modelo ainda é válido, e as características descritas para cada camada ainda são importantes, apesar dos protocolos associados ao modelo raramente serem utilizados (TANENBAUM; WETHERALL, 2011). A Figura 13 ilustra o modelo.

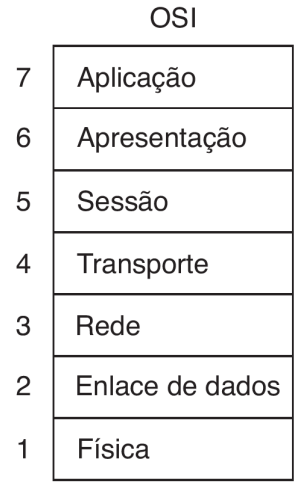

Figura 13: Modelo de referência OSI

Fonte: Tanenbaum e Wetherall (2011)

A Figura 13 apresenta as sete camadas do modelo OSI, sendo que a camada mais baixa é a camada física, e a mais alta é a camada de aplicação. Uma breve descrição de cada uma delas se segue (KUROSE; ROSS, 2013; TANENBAUM; WETHERALL, 2011).

- A camada de aplicação contém vários protocolos úteis aos usuários, como aqueles para transferência de arquivos ou correio eletrônico. Um exemplo de protocolo dessa camada é o *Hypertext Transfer Protocol* (HTTP), utilizado quando um navegador solicita uma página da Internet ao servidor que hospeda essa página.

- A camada de apresentação lida com sintaxe e semântica das informações transmitidas. Ela permite a comunicação entre computadores com diferentes representações internas de dados, além de definir e realizar o intercâmbio de estruturas de dados de nível mais alto.

- A camada de sessão permite que usuários em diferentes máquinas estabeleçam sessões de comunicação entre eles, mantendo o controle de quem transmite em qual momento, e realizando uma verificação periódica de longas transmissões, assim permitindo que continuem do ponto que estavam na ocorrência de alguma falha, com subsequente recuperação.

- A camada de transporte aceita dados da camada de sessão, os divide em unidades menores caso necessário, e as repassa à camada de rede, garantindo que todos os fragmentos cheguem corretamente, e de forma eficiente, à outra extremidade. Isso deve ser feito de forma que as camadas superiores fiquem isoladas a mudanças de *hardware*.

- A camada de rede é responsável por definir as rotas e a maneira como os pacotes vão sair da origem e chegar no destino. Questões como qualidade de serviço (atraso, tempo em trânsito, etc.) e controle de congestionamento também são de responsabilidade dessa camada, além de permitir que redes heterogêneas, com diferentes protocolos e endereçamentos, sejam interconectadas.

- A camada de enlace basicamente move um datagrama[1] de um nó até um nó adjacente por um único enlace de comunicação. Nó é qualquer dispositivo que execute um protocolo da camada de enlace, podendo ser um equipamento como um *laptop*, ou roteadores, pontos de acesso Wi-Fi e comutadores. Por outro lado, é dado o nome de enlace ao canal de comunicação que conecta nós adjacentes. Alguns detalhes do serviço da camada de enlace podem mudar, dependendo do protocolo. Uma das funções dessa camada é encapsular cada datagrama em um quadro da camada de enlace, além de garantir entrega confiável e possuir mecanismos de detecção e correção de erros implementados em *hardware*. A estrutura desse quadro muda, dependendo do protocolo utilizado.

- A função da camada física é movimentar os *bits* individuais que estão dentro dos quadros de um nó para outro. Os protocolos dessa camada vão depender do meio de transmissão, já que em cada meio, o *bit* atravessa o enlace de uma maneira diferente.

Dada as definições dos serviços de cada camada do modelo OSI, será mostrado a seguir um modelo mais antigo: o modelo TCP/IP. Definido pela primeira vez no início da década de 1970 motivado pela limitação dos protocolos utilizados na época pela ARPANET, a rede de pesquisa patrocinada pelo Departamento de Defesa dos EUA, que mais tarde viria a se tornar a Internet que se conhece hoje (TANENBAUM; WETHERALL, 2011).

O modelo OSI descreve bem o serviço de cada camada, mas seus protocolos raramente são utilizados. Já o modelo TCP/IP possui características opostas: o modelo é pouco usado, mas seus protocolos ainda são muito utilizados (TANENBAUM; WETHERALL, 2011). A figura 14 ilustra o modelo TCP/IP:

A Figura 14 mostra as quatro camadas do modelo de referência TCP/IP. A camada 1 (Acesso à Rede) engloba as camadas física e de enlace do modelo OSI. As camadas 2 e 3 são análogas ao modelo OSI, enquanto a quarta e última camada

[1] **Pacotes de dados da camada de rede (KUROSE; ROSS, 2013).**

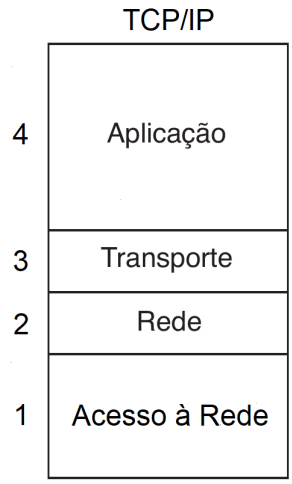

Figura 14: Modelo de Referência TCP/IP

engloba as camadas de sessão, apresentação e aplicação do modelo OSI. A Figura 15 ilustra alguns protocolos de cada camada do modelo TCP/IP.

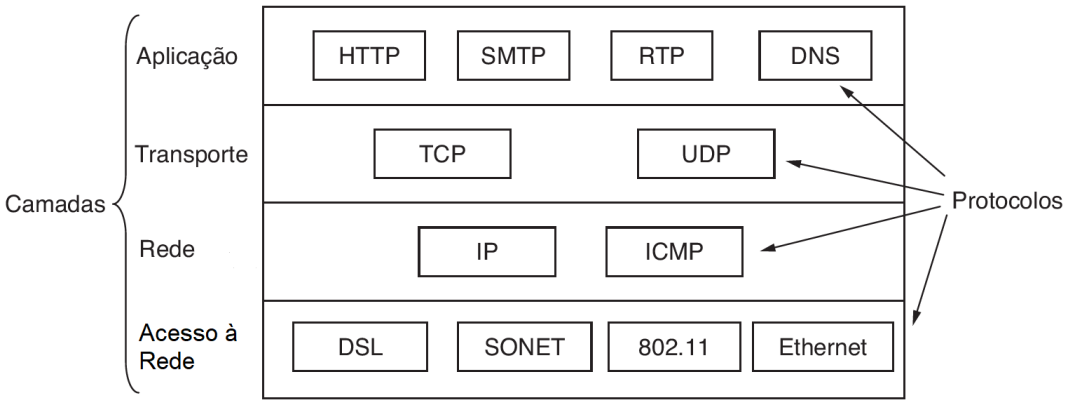

Figura 15: Exemplos de protocolos de cada camada do modelo TCP/IP

Fonte: Adaptado de (TANENBAUM; WETHERALL, 2011)

Na Figura 15, dois protocolos da camada de transporte estão ilustrados: o *Transport Control Protocol* (TCP) e o *User Datagram Protocol* (UDP). O TCP é um protocolo orientado a conexões confiável, que garante a entrega ordenada e sem erros de um fluxo de *bytes*. Além disso, também efetua controle de fluxo, isso é, impede que um transmissor rápido sobrecarregue um receptor lento. É o protocolo ideal quando a entrega tem que ser garantida, mas devido ao maior número de informações de controle, não é recomendado para aplicações que precisam de entrega imediata. Porém, se entrega imediata for o fator dominante, o protocolo ideal é o UDP. É um protocolo não orientado a conexões e não confiável, utilizado em aplicações como transmissão de voz ou vídeo em tempo real (TANENBAUM; WETHERALL, 2011).

Na camada de Rede, há o *Internet Protocol* (IP), que é um protocolo responsável pelas funções de endereçamento e repasse na Internet. Por fim, a camada de Acesso à Rede da Figura 15 ilustra alguns protocolos equivalentes à camada de enlace do modelo OSI: Ethernet e IEEE 802.11. Enquanto a unidade de medição sincrofasorial de Grando (2016) utiliza o protocolo Ethernet para envio de dados via conexão cabeada, o sistema embarcado ESP32WROOM utilizado nesse trabalho utilizará o protocolo IEEE 802.11 como protocolo de enlace, também conhecido como Wi-Fi.

2.5 WI-FI - PROTOCOLO 802.11

O protocolo 802.11, ou Wi-Fi, é um padrão definido pela IEEE para redes locais sem fio. Opera nas bandas não licenciadas de frequência Industrial, Científico e Médico (ISM), como 902-928MHz, 2,4-2,5GHz ou 5,725-5,825GHz. Essas bandas foram escolhidas para que não fosse necessário licenciar um espectro de frequência para esse fim, evitando o alto custo consequente. Todos os dispositivos podem usar esse espectro, desde que limitem sua potência de transmissão para permitir a coexistência (TANENBAUM; WETHERALL, 2011). Os elementos básicos de uma rede sem fio são: hospedeiro sem fio, enlace sem fio e estação-base (KUROSE; ROSS, 2013):

- Os hospedeiros, também conhecidos como sistemas finais, são os equipamentos que executam aplicações, como *laptops* ou *smartphones*.
- Os enlaces, assim como já explicados na seção anterior, são canais de comunicação entre nós adjacentes. Diferentes tecnologias de enlace sem fio possuem taxas de transmissão e áreas de cobertura variadas com o qual operam.
- Estação-base, também conhecida como Ponto de Acesso (AP), é uma peça chave na infraestrutura de rede sem fio. Ela é responsável por envio e recepção de dados de um hospedeiro sem fio associado a ela. Como comunicação sem fio se baseia na difusão de ondas de rádio em um meio compartilhado, é esperado que hajam múltiplos usuários associados a uma mesma estação-base, sendo também função dela coordenar a transmissão desses vários hospedeiros sem fio. Quando dito que um hospedeiro sem fio está associado a uma estação-base, significa que esse hospedeiro está dentro do alcance de comunicação sem fio do ponto de acesso, e que esse hospedeiro usa a estação-base para retrans-

mitir dados entre ele e uma rede maior, que pode ser a Internet ou uma rede corporativa.

Quando hospedeiros estão associados a uma estação-base, é dito que operam em modo de infraestrutura, pois serviços tradicionais de rede, como roteamento e atribuição de endereços, são fornecidos pela rede maior conectada a estação-base acessada pelo hospedeiro sem fio. Quando não há essa infraestrutura e os hospedeiros sem fio conectam-se diretamente uns aos outros, dá-se o nome de rede *ad-hoc* (KUROSE; ROSS, 2013). Neste trabalho, a discussão será focada no modo de infraestrutura, pois é dessa forma que o sistema embarcado ESP32WROOM irá operar. A Figura 16 ilustra esses dois modos de operação.

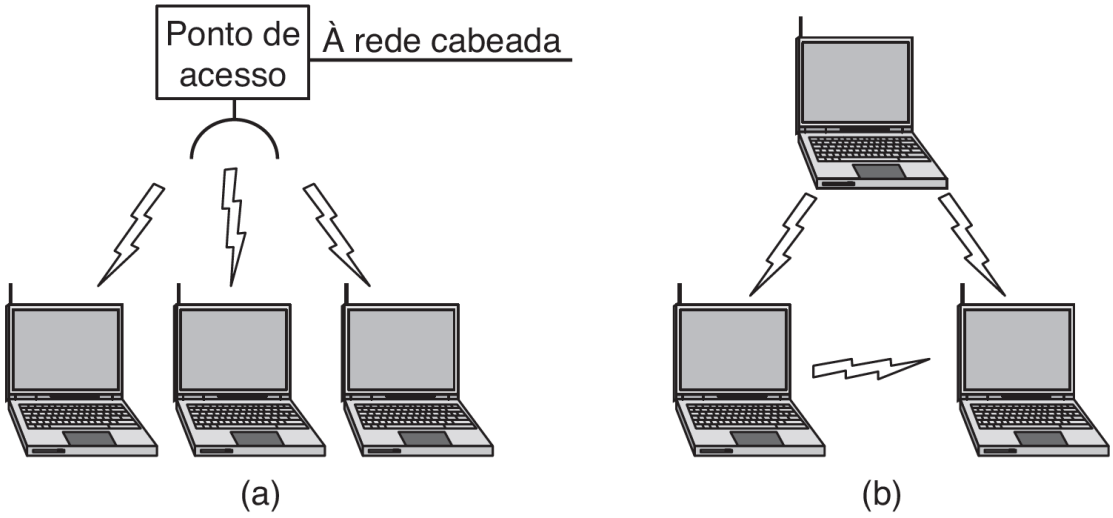

Figura 16: a) Rede sem fio com ponto de acesso. b) Rede *ad-hoc*

Fonte: Tanenbaum e Wetherall (2011)

Há diversos padrões IEEE 802.11, entre eles o 802.11b, 802.11g ou o 802.11n. Cada um deles pode possuir uma frequência de operação, largura de banda e técnica de modulação[1] diferentes, o que por consequência altera sua taxa de transmissão e alcance efetivo de operação. Normalmente, os hospedeiros sem fio e os pontos de acesso operam com mais de um desses padrões para garantir compatibilidade, como é o caso do sistema embarcado utilizado nesse trabalho, que suporta os padrões 802.11 b/g/n. Um breve resumo de alguns desses padrões é ilustrado na Tabela 2 (KUROSE; ROSS, 2013; KAEWKIRIYA, 2017).

Como dito anteriormente, uma das funções da estação-base é coordenar as transmissões de múltiplos hospedeiros sem fio associados a ele, para evitar colisões.

[1]**Processo no qual a informação a ser transmitida é adicionada a ondas de rádio**

Tabela 2: Padrões IEEE 802.11

Padrão	Faixa de Frequências (EUA)	Taxa de Dados
802.11b	2,4 - 2,485 GHz	Até 11 Mbits/s
802.11a	5,1- 5,8 GHz	Até 54 Mbits/s
802.11g	2,4 - 2,485 GHz	Até 54 Mbits/s
802.11n	2,4 - 5 GHz	Até 300 Mbits/s
802.11ac	5 GHz	Até 1,35 Gbits/s

O protocolo de acesso ao meio utilizado para redes locais 802.11 é o *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). O funcionamento básico do CSMA é que ao tentar transmitir, um nó primeiramente escuta o meio de transmissão para determinar se alguma transmissão está ocorrendo. Em caso afirmativo, o nó aguarda até a transmissão estar completa antes de iniciar sua própria. A prevenção de colisão utilizada no 802.11 (e a subsequente sigla CSMA/CA) é que se alguma transmissão em andamento for detectada, o nó que deseja transmitir aguarda por um tempo aleatório antes de tentar novamente, diminuindo a chance de dois nós começarem a transmitir ao mesmo tempo após a transmissão anterior ter sido finalizada (KUROSE; ROSS, 2013).

2.6 COMUNICAÇÃO ENTRE PROCESSOS UTILIZANDO SOCKETS

A Seção 2.5 teve por foco um dos protocolos da camada de enlace, assim como alguns detalhes pertinentes à camada física. Nesta seção, o foco será na interface de *software* entre a camada de aplicação e de transporte, conhecida como *socket*, para comunicação entre processos executados em sistemas finais diferentes.

Um processo é uma instância de um programa em execução, incluindo os valores atuais do contador de programa, registradores e variáveis (TANENBAUM; WOODHULL, 2006). Processos em dois sistemas finais diferentes se comunicam pela troca de mensagens através da rede de computadores. Para cada par de processos em comunicação, é comumente dado o nome de cliente a um deles e servidor ao outro. Cliente é aquele que inicia a comunicação, e servidor é o que espera ser contatado para iniciar a sessão (KUROSE; ROSS, 2013).

Um processo envia e recebe mensagens da rede através de um *socket*. Uma analogia da relação entre processo e *socket* é imaginar o processo como sendo uma

casa, e o *socket* como sendo a porta dessa casa. Quando o processo quer enviar uma mensagem para um processo em outro sistema final, ele passa a mensagem pela porta (que é o *socket*) e uma infraestrutura de transporte leva essa mensagem até a porta do outro processo, que concebe alguma ação sobre a mensagem (KUROSE; ROSS, 2013). Dessa maneira, o desenvolvedor tem total controle sobre o que está no lado da camada de aplicação do *socket*, mas pouco controle sobre o que está no lado da camada de transporte. Uma das poucas decisões feitas pelo desenvolvedor sobre o lado da camada de transporte é qual protocolo utilizar: TCP ou UDP. A figura 17 ilustra a comunicação entre dois processos utilizando *sockets* TCP.

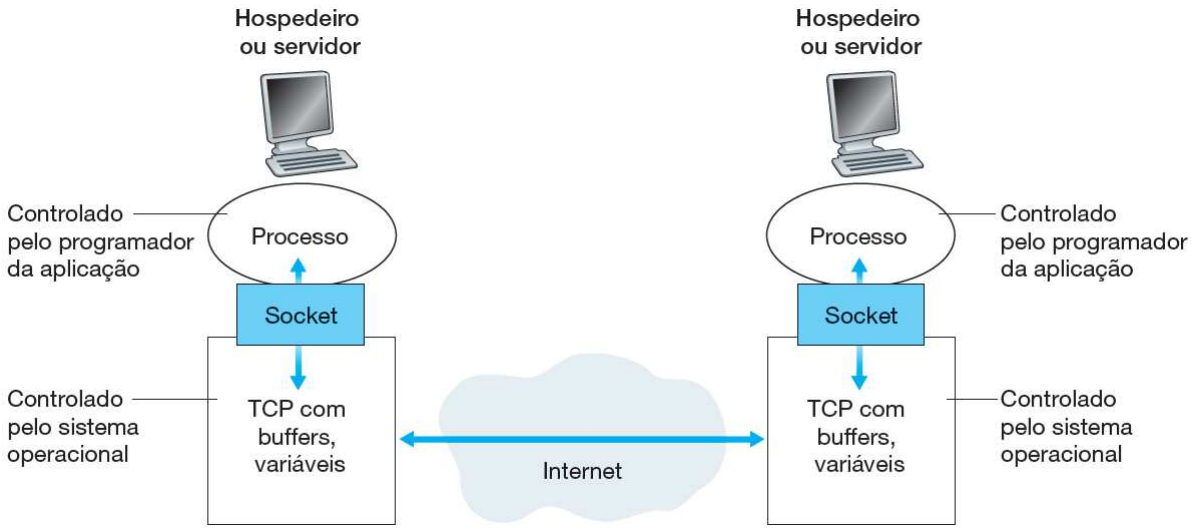

Figura 17: Troca de mensagens entre dois processos por meio de *sockets* TCP

Fonte: Kurose e Ross (2013)

A forma de se trabalhar com *sockets* muda, dependendo do protocolo da camada de transporte escolhido. Em *sockets* UDP, cada pacote enviado precisa conter o endereço IP e número de porta do *socket* de destino, assim como as mesmas informações do remetente. O endereço IP é utilizado para rotear o pacote do emissor pela Internet até o receptor, e o número de porta é o identificador designado ao *socket*, para que seja possível distingui-lo dos demais *sockets* de outras aplicações de rede executando no mesmo sistema final (KUROSE; ROSS, 2013).

Diferente do UDP, o protocolo TCP é orientado a conexões. Isso significa que em um *socket* TCP, é estabelecida uma conexão entre o cliente e o servidor. Uma ponta dessa conexão está ligada ao *socket* do cliente e a outra está ligada ao *socket* do servidor. Ao criar a conexão TCP, é associada a ela o endereço de *socket* (IP e número de porta) tanto do cliente quanto do servidor. Depois de estabelecida a conexão, tudo que um lado precisa fazer para enviar dados ao outro é inseri-los na

conexão TCP de seu *socket*. Isso é diferente do socket UDP, em que as informações de endereço de *socket* precisam estar contidas em cada um dos pacotes enviados (KUROSE; ROSS, 2013).

Estabelecido qual *socket* utilizar, é necessário um protocolo para definir os métodos para troca de dados em tempo real entre a unidade de medição sincrofasorial e o PDC. Isso está previsto na norma IEEE C37.118.2 - 2011, que define que os métodos previstos de comunicação IP são TCP, UDP e TCP/UDP, além de normatizar as mensagens que podem ser utilizadas, bem como o formato dos *frames* de dados, definindo seus campos e seus respectivos tamanhos em *bytes* (IEEE, 2011a).

3 MATERIAIS E MÉTODO

3.1 INTRODUÇÃO

Este capítulo é responsável por apresentar os materiais e *softwares* a serem utilizados neste trabalho, bem como a metodologia de desenvolvimento. Para melhor visualização das etapas a serem desenvolvidas, será antes realizada uma breve introdução de qual etapa da PMU de Grando (2016) será alterada e o que este trabalho visa adicionar. A Figura 18 é uma adaptação da arquitetura proposta por Grando (2016) na Figura 7 e apresenta a atual estrutura de sua PMU.

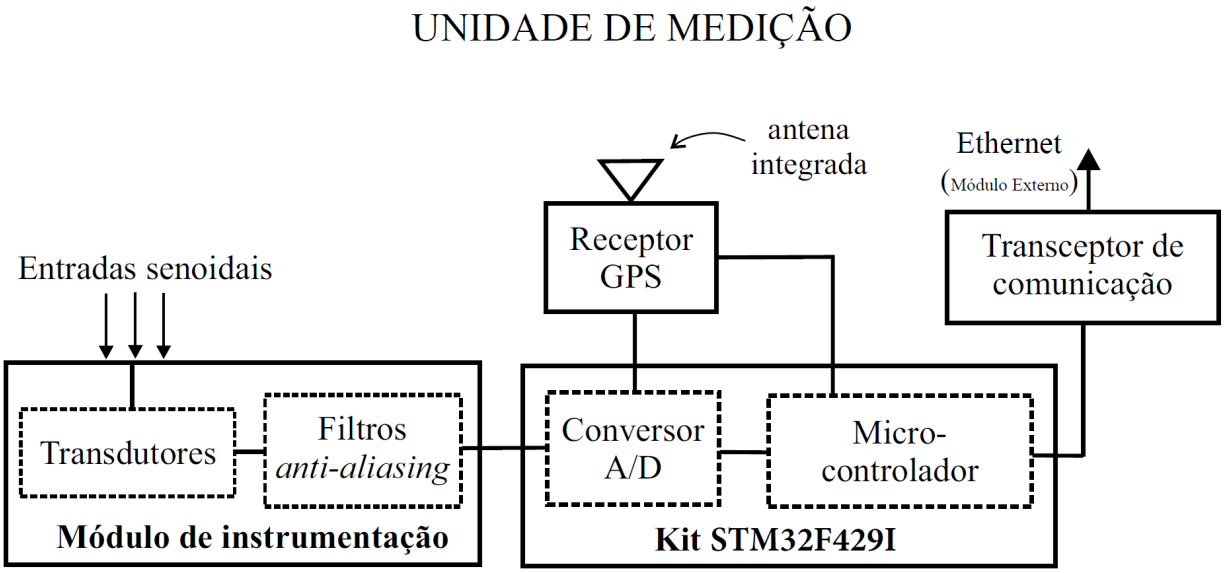

Figura 18: Atual estrutura da PMU de Grando

Fonte: Adaptado de Grando (2016)

Conforme a Figura 18, o *kit* atualmente utilizado na PMU de Grando (2016) é o STM32F429I-*Discovery*, que não possui módulo Ethernet no próprio *kit*, sendo necessária a utilização de um módulo externo para que seja possível enviar os dados coletados a um PDC por meio de conexão cabeada com a Internet. A primeira etapa deste trabalho é realizar a migração do *kit* STM32F429I-*Discovery* para o *kit* STM32F746G-*Discovery*, que já possui módulo *Ethernet* próprio. A Figura 19 apresenta a estrutura da PMU após a realização desta etapa.

Após realizada a migração do *kit* utilizado na PMU, será estabelecido um barramento de comunicação SPI entre a PMU e o sistema embarcado ESP32WROOM. Dessa forma, o ESP32, que possui suporte a Wi-Fi, poderá receber os dados coletados pela PMU e efetuar o envio desses dados a um PDC por meio de conexão sem-fio com a Internet. A Figura 20 apresenta o sistema finalizado proposto.

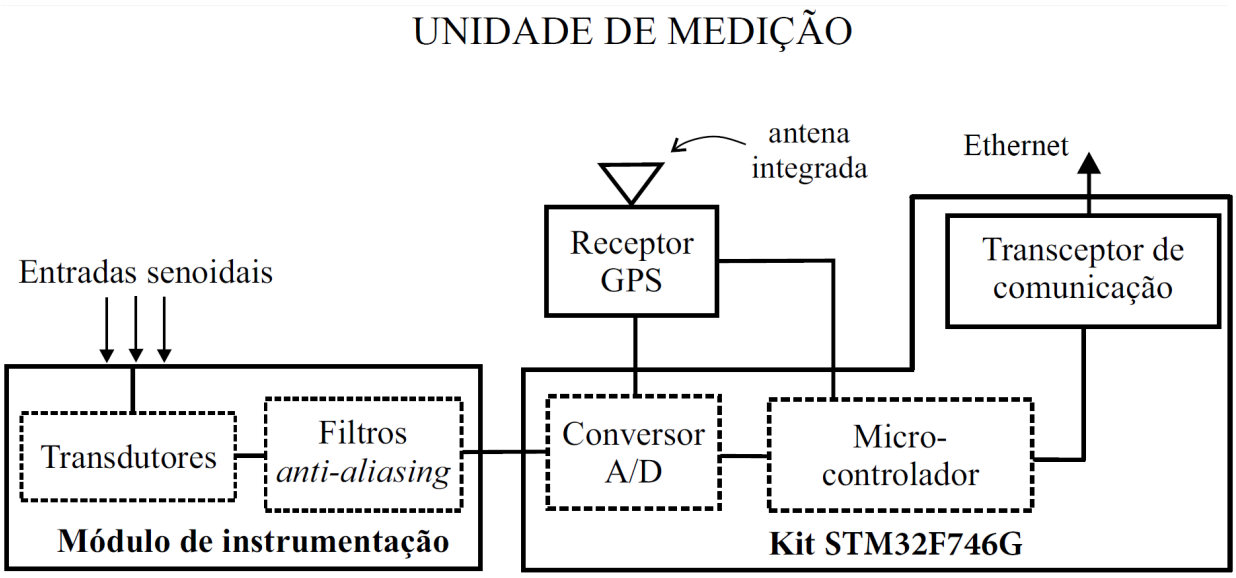

Figura 19: Estrutura da PMU após a migração para outro *kit*

Fonte: Adaptado de Grando (2016)

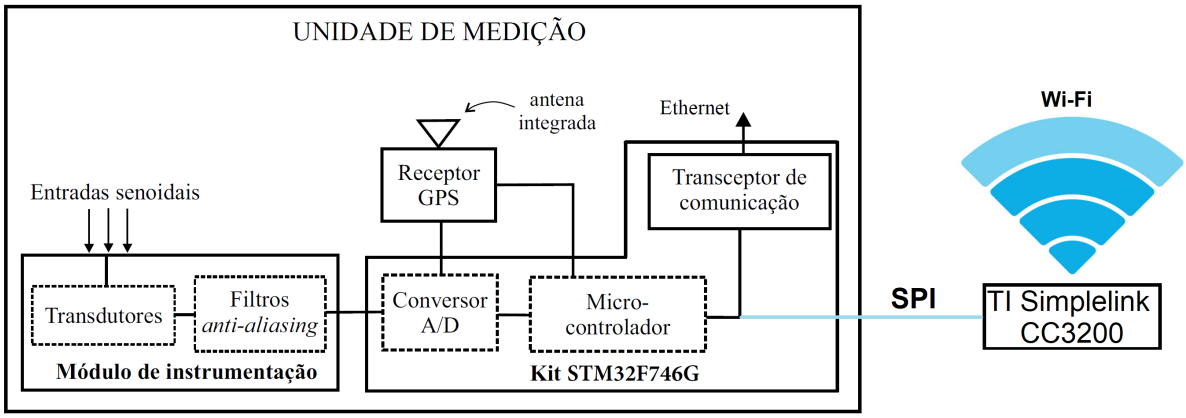

Figura 20: Sistema finalizado proposto

3.2 MATERIAIS

3.2.1 ESP32 WROOM

Inicialmente foi proposto a utilização do sistema embarcado Simplelink CC3200 da Texas Instruments para realizar a transmissão de dados fasoriais ao PDC por meio de conexão sem-fio com a internet, o que iria envolver a necessidade de utilização do RTOS da Texas Instruments (o TI-RTOS) e o aprendizado de utilização de uma nova pilha de protocolos TCP/IP.

Uma alternativa viável para o mesmo propósito é o ESP32 Wroom, da Espressif Systems, um *kit* de desenvolvimento de IoT com custo aproximado de 2 dólares que possui suporte a Wi-Fi b/g/n, Bluetooth, Bluetooth Low Energy e periféricos como UART, SPI, SDIO, I2C ou I2S. Possui um *clock* de 240MHz e 4MB de memória *flash*.

Tudo isso aliado a utilização de FreeRTOS e a pilha de protocolos lwIP faz com que o tempo e complexidade de desenvolvimento da última etapa deste trabalho seja reduzida significativamente. Devido a esses fatores, optou-se por utilizá-lo ao invés do Simplelink CC3200 da Texas Instruments. A Figura 21 apresenta o *kit* ESP32 Wroom.

Figura 21: ESP32 Wroom

Fonte: ALIEXPRESS (2018)

3.2.2 STM32F746G DISCOVERY KIT

Um dos *kits* de desenvolvimento utilizado neste trabalho, o qual irá substituir o *kit* STM32F429I-*Discovery* originalmente utilizado por Grando (2016) em sua PMU. A Tabela 3 apresenta um comparativo entre as características dos dois *kits* e seus respectivos microcontroladores, seguida de imagem frontal e traseira do *kit* STM32F746G-*Discovery* nas Figuras 22 e 23, respectivamente (STMICROELECTRONICS, 2016a; STMICROELECTRONICS, 2016b; STMICROELECTRONICS, 2015; STMICROELECTRONICS, 2016c).

3.2.3 IDE STM32 CUBEMX

Um Ambiente de Desenvolvimento Integrado (IDE) desenvolvida pela STMicroelectronics, para auxiliar o desenvolvedor na criação de aplicações usando seus microcontroladores. Nela é possível configurar de forma gráfica todos os periféricos e *clocks*, e a IDE gera o código de configuração em linguagem C da plataforma escolhida

pelo desenvolvedor. Dessa forma, há uma redução considerável do tempo necessário para desenvolver a aplicação, pois o desenvolvedor pode focar-se em cuidar do que a aplicação deve fazer, e não em configurar a plataforma via código (STMICROELECTRONICS, 2017).

Tabela 3: Comparativo entre o *kit* originalmente utilizado na PMU e o *kit* a ser migrado

Características	***Kit* STM32F429I-*Discovery***	***Kit* STM32F746G-*Discovery***
Microcontrolador	STM32F429ZIT6 com ARM Cortex-M4	STM32F746NGH6 com ARM Cortex-M7
Frequência de *clock* (MHz)	180	216
DMIPS	225	462
DMIPS/MHz	1,25	2,14
Memória FLASH (MB)	2	1
Memória RAM (KB)	256	320
SDRAM para a tela (*Mbits*)	64	128
Conversores A/D	3 conversores independentes, 24 canais	3 conversores independentes, 24 canais
Resolução de Conversores A/D (*bits*)	12	12
Conversores D/A	2	2
Resolução de Conversores D/A (*bits*)	12	12
Interfaces de Comunicação Suportadas pelo Microcontrolador	I2C, USART/UART, SPI, I2S, SAI, CAN 2.0B, SDIO, USB 2.0 OTG, Ethernet MAC 10/100	I2C, USART/UART, SPI, I2S, SAI, CAN 2.0B, SDMMC, SPDIFRX, HDMI-CEC, USB 2.0 OTG, Ethernet MAC 10/100
Módulo Ethernet no próprio *kit*	Não	Sim
Display LCD sensível ao toque	2,4 polegadas	4,3 polegadas

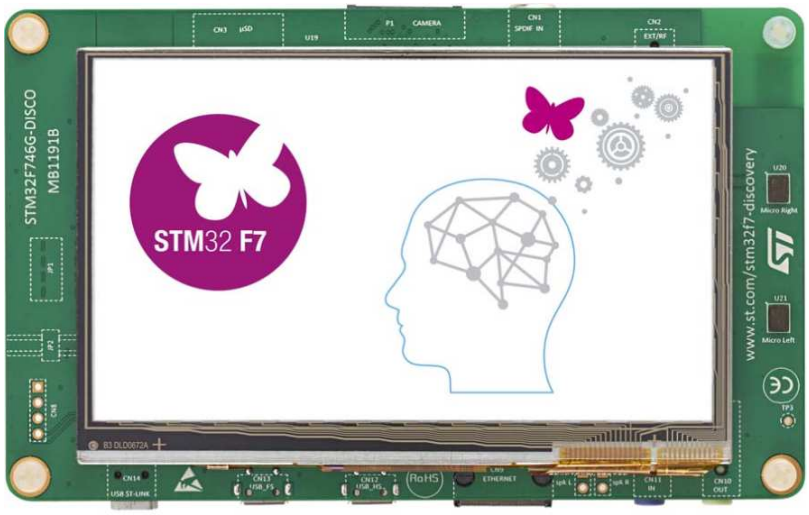

Figura 22: Parte frontal do *kit* STM32F746G

Fonte: STMICROELECTRONICS (2015)

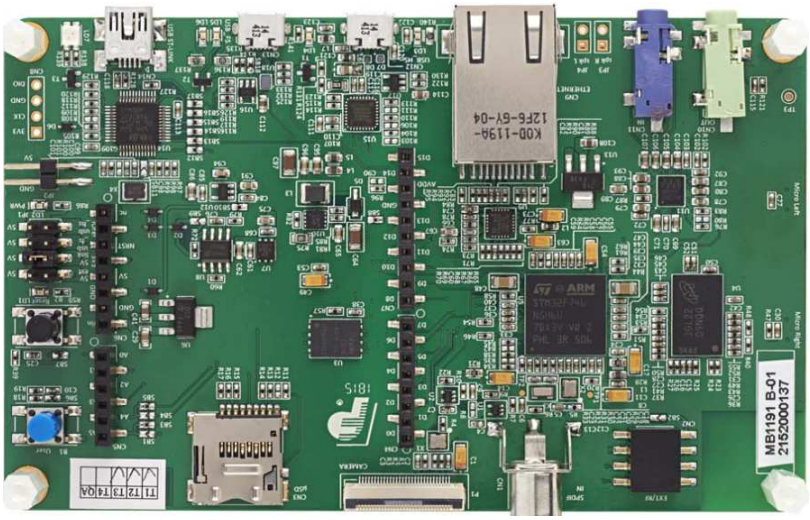

Figura 23: Parte traseira do *kit* STM32F746G

Fonte: STMICROELECTRONICS (2015)

3.3 MÉTODO

As etapas do desenvolvimento deste trabalho foram definidas da seguinte forma:

- Migrar o código desenvolvido por Grando (2016) em sua PMU do *kit* STM32F429I-*Discovery* para o *kit* STM32F746G-*Discovery*, ou seja, definir quais portas de entrada e saída serão utilizados no *kit* a ser migrado e ajustar o código de configuração dos periféricos e interfaces de comunicação necessários.

- Realização de testes para verificar se a PMU, agora com o *kit* STM32F746G-*Discovery*, está realizando corretamente a estimação fasorial e enviando os dados a um PDC por meio de conexão cabeada com a Internet.
- Estabelecer um barramento de comunicação SPI entre a PMU e o ESP32WROOM, definindo um protocolo SPI comum a ambos os sistemas embarcados e implementando o código de configuração necessário nas duas plataformas. Esse barramento será utilizado para enviar os dados coletados pela PMU ao ESP32WROOM.
- Com o barramento de comunicação SPI entre os dois sistemas embarcados estabelecido, implementar no ESP32WROOM o envio dos dados recebidos pela PMU a um PDC por meio de conexão sem fio com a Internet.
- Realização de testes para verificar se os dados fasoriais estão sendo corretamente transmitidos ao PDC por meio de conexão sem-fio com a Internet.

4 DESENVOLVIMENTO

Neste capitulo serão apresentadas as etapas de implementação deste trabalho. A seção 4.1 irá abordar a migração do código da PMU de Grando (2016) do *kit* STM32F429I-*Discovery* para o *kit* STM32F746GI-*Discovery*, enquanto a seção 4.2 apresentará a adição da funcionalidade de enviar dados fasoriais para o PDC por meio de conexão sem-fio com a internet, Wi-Fi.

4.1 MIGRAÇÃO DE CÓDIGO

Essa etapa da implementação envolveu identificar quais elementos do código da PMU precisariam ser modificados e quais poderiam ser mantidos. Após inspeção do código-fonte da PMU de Grando (2016) e das funcionalidades do *kit* STM32F746G-*Discovery*, foi verificado que poderia ser utilizada a mesma pilha de protocolos TCP/IP para transmissão de dados fasoriais para o PDC, mas que seria necessário utilizar um outro conjunto de *drivers* para configuração e utilização dos periféricos, assim como um outro Sistema Operacional de Tempo Real (RTOS)[2]. Um comparativo entre os aspectos de implementação da PMU de Grando (2016) e o *kit* STM32F746G-*Discovery* é apresentado na tabela 4.

Tabela 4: Comparativo entre aspectos de implementação dos dois sistemas embarcados envolvidos no *port*

Características	**PMU de Grando (2016)**	***Kit* STM32F746G-Discovery**
***Drivers* de Periféricos**	*Standard Peripheral Library*	STCube HAL
RTOS	BRTOS	*Middleware* CMSIS-RTOS + FreeRTOS
Pilha de protocolo TCP/IP	lwIP	lwIP

Na realização dessa etapa, o código da aplicação da PMU de Grando (2016) permanece inalterado e com a mesma estrutura de projeto, sendo necessário apenas

[2]**RTOS é um sistema operacional capaz de oferecer a noção de multitarefa enquanto garante um padrão de resposta que se mantém dentro de restrições de tempo (NOVIELLO, 2016).**

alterar as chamadas de funções da API dos periféricos, bem como a configuração e utilização de rotinas específicas do RTOS, como filas de mensagem, semáforos, mutexes, criação de tarefas e *delays*.

Em relação aos *drivers* de periféricos, a biblioteca oficial de desenvolvimento de aplicações para os microcontroladores STM32 se tornou o STCube HAL (NOVIELLO, 2016), que substituiu a *Standard Peripheral Library*, biblioteca presente no código da PMU de Grando (2016), conforme Tabela 4. Para realizar essa mudança de APIs dos periféricos, foi utilizada a documentação oficial do STCube HAL, bem como bibliografia dedicada aos microcontroladores STM32, como o livro *Mastering STM32* (NOVIELLO, 2016).

Uma das mudanças identificadas entre as bibliotecas de periféricos é na utilização de rotinas de tratamento de interrupção (RTI). Os *drivers* dos periféricos possuem uma função *handler* que podem ser utilizadas dentro das RTIs dos respectivos periféricos. Essa função lida com detalhes que normalmente caberiam ao programador realizar, como limpar *flags* de interrupção e outras tratativas. Após sua execução, são chamadas funções de *callback* para situações específicas, e são nelas em que a aplicação faz o tratamento dos dados.

Ao tomar como exemplo o *driver* do periférico UART, há mais de uma situação diferente em que sua RTI será chamada, como término de recepção ou transmissão de dados. Com o STMCube HAL, a função *handler* pode ser chamada dentro da RTI, e haverá funções de *callback* específicas para transmissão ou recepção de dados, conforme o trecho de código a seguir exemplifica.

```
/* Rotina de tratamento de interrupcao de uma interface UART */
void USART_IRQHandler(void){
  /* Funcao handler do driver da UART do STCube HAL */
  HAL_UART_IRQHandler(&huart);
}

/* Callback de transmissao completada */
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart){
 /* Tratativa por parte da aplicacao apos os dados serem
    transmitidos */
}

/* Callback de recepcao completada */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
 /* Tratativa por parte da aplicacao apos os dados serem
```

```
recebidos */
}
```

Quanto ao RTOS, a PMU de Grando (2016) utilizou o BRTOS, um RTOS brasileiro desenvolvido por Gustavo Weber Denardin e Carlos Henrique Barriquello. No código migrado foi utilizado o FreeRTOS, que além de ser um dos RTOS mais utilizados no mundo, também é gratuito. Porém, ele não foi utilizado diretamente. A ST aderiu ao Padrão de Interface de *Software* de Microcontroladores Cortex (CMSIS), uma camada de abstração de *hardware* independente de fabricante idealizada pela ARM, com o objetivo de aumentar a portabilidade de aplicações entre RTOS e microcontroladores de diferentes fabricantes utilizando um conjunto padronizado de APIs (NOVIELLO, 2016). Por esse motivo, o código relacionado a chamadas e funções do RTOS foi substituída do BRTOS para a API do CMSIS, e essa por sua vez realiza as chamadas de funções do FreeRTOS.

Com os aspectos de implementação analisados, foi verificado quais periféricos deveriam ser configurados para realizar a migração de código. São eles:

- Configuração de *timer* acionado por fonte externa;
- Configuração de três conversores analógicos-digitais com conversão simultânea acionada pelo *timer*;
- Configuração de interface UART;
- Configuração de módulo Ethernet nativo do *kit* STM32F746G-*Discovery*.

A configuração do *timer* acionado por fonte externa, que por sua vez é o gatilho para iniciar a conversão dos três conversores analógico-digitais, é utilizada para que o pulso PPS proveniente do GPS inicie a conversão das três fases do sistema elétrico simultaneamente, para que a estimação fasorial implementada por Grando (2016) possa ser realizada.

A configuração da interface UART foi realizada para que o GPS informe a estampa de tempo utilizada nos *frames* de dados enviados ao PDC. Foi instalada uma tarefa específica para o GPS, em que nela é iniciada a recepção de dados. Sempre que uma mensagem é recebida, é gerada uma interrupção que direciona o fluxo de execução de código para um *callback* de recepção. Os dados são inseridos em uma fila de mensagens do RTOS, e ao ler os dados recebidos dessa fila, é realizada a tratativa da mensagem recebida para identificar a estampa de tempo, que é um número inteiro representando a quantidade de segundos desde 1 de janeiro de 1970. O

protocolo das mensagens enviadas pelo GPS é o NMEA (*National Marine Electronics Association*).

A configuração da interface Ethernet foi realizada a partir de um código de exemplo, mas todo o código relacionado às camadas de rede acima da camada de enlace se manteve inalterado, devido ao fato de ser utilizada a mesma pilha de protocolo TCP/IP, a lwIP.

4.2 TRANSMISSÃO POR WI-FI

Com as etapas anteriores realizadas, a migração de código foi concluída. Porém, para que o envio de dados ao PDC por meio de conexão sem-fio com a Internet possa ser realizado, é necessário enviar os dados estimados para outro microcontrolador. Para esse propósito, foi configurada a interface SPI como dispositivo mestre. Ao término de cada estimação fasorial, é preparado um vetor de dados para transmissão SPI com as informações necessárias a serem transmitidas para que o outro microcontrolador possa montar os *frames* dados para envio ao PDC. A tabela 5 apresenta quais informações são transmitidas e em qual ordem.

Tabela 5: Informações enviadas por SPI

Informação	**Quantidade de *bytes***
Estampa de tempo	4
Fração de segundo	4
Magnitude (R)	4
Fase (R)	4
Magnitude (S)	4
Fase (S)	4
Magnitude (T)	4
Fase (T)	4
Frequência	4
Variação de Frequência	4
Dummy	3
Checksum	1

O *Checksum* é composto pelo *byte* menos significativo da soma entre todos os outros 43 *bytes*. Os bytes Dummy foram incluídos devido a limitações no projeto da API no microcontrolador utilizado para conexão Wi-Fi: a quantidade de *bytes* em uma transmissão SPI deve ser maior que 8 e divisível por 4 para que não hajam perdas na transmissão (ESPRESSIF, 2018).

Com as ações citadas anteriormente concluídas, a próxima parte da implementação foi com o sistema embarcado que suporta a transmissão sem-fio pela internet, o ESP32 Wroom.

Para a realização dessa etapa no ESP32 Wroom, foi utilizado o *Espressif IoT Development Framework*, um conjunto de ferramentas para desenvolvimento de aplicações utilizando *chips* ESP32. Foi utilizado o sistema operacional Ubuntu para utilização desse *framework* por meio de uma máquina virtual. O *framework* vem com diversos exemplos de código de configuração e utilização de periféricos, além de possuir um guia oficial de referência.

A configuração SPI do ESP32 foi realizada com base em um desses exemplos, e ele foi definido como dispositivo escravo. Uma tarefa foi instalada especificamente para receber transmissões SPI do dispositivo mestre. Para evitar que as transmissões por parte do STM32F746G-Discovery iniciem antes que o ESP32 esteja pronto, foi definido um GPIO em cada sistema embarcado para atuar como linha de *handshake*. O estado inativo dessa linha é em nível alto, e quando o dispositivo escravo está pronto para receber uma transmissão, ele coloca essa linha em nível baixo, gerando uma borda de descida no dispositivo mestre, que está configurado para gerar uma interrupção que irá liberar a transmissão SPI das estimações fasoriais.

O *driver* de SPI como dispositivo escravo possui duas funções de utilização: uma bloqueante e outra não bloqueante. A bloqueante é chamada por meio da função `spi_slave_transmit()` e apenas retorna quando uma transmissão por parte do mestre é completada. Já a função `spi_slave_queue_trans()` é não bloqueante e coloca uma transação na fila de espera, com o máximo de transações simultâneas sendo configurável pelo *driver* do dispositivo. Quando uma transmissão por parte do mestre é completada, uma função de *callback* é chamada e é necessário utilizar a rotina `spi_slave_get_trans_result()` para receber o conteúdo da transmissão por parte do dispositivo mestre. Optou-se por utilizar a função bloqueante para que o processador fosse liberado enquanto não houvessem dados a serem recebidos.

Após realizadas as configurações de transmissão SPI, foram utilizados exemplos de configuração da interface Wi-Fi para realizar a última etapa de desenvolvi-

mento deste trabalho. O SSID da rede e a senha são configuradas no código-fonte e passadas via parâmetro para a API, que realiza a conexão e adquire um endereço IP dinâmicamente por meio de DHCP. Apenas a partir desse ponto é possível estabelecer uma conexão *socket* TCP com outra aplicação, já que é necessário um par IP + Porta para tal, sendo que o segundo elo do par é configurado via código-fonte.

Devido ao ESP32 possuir suporte a pilha de protocolos lwIP, foi possível reutilizar o código de envio de dados fasoriais ao PDC presente no código migrado da PMU. Quando o endereço IP é adquirido por meio de DHCP, é instalado uma tarefa para atender as requisições do PDC, visto que a PMU atua como servidor *socket*.

A tarefa `taskSPI` é responsável pelas instruções relacionadas ao SPI. Ao receber uma transmissão de dados, eles são copiados para um vetor auxiliar e o *checksum* é calculado. Em seguida, a liberação de um semáforo indicando que a transmissão de dados ao PDC pode ser realizada é efetuada quando três condições são satisfeitas:

- O semáforo já foi instanciado;
- A conexão *socket* com o PDC já foi estabelecida e a aplicação está liberada a enviar o *frame* de dados;
- O *checksum* está correto.

Após liberado o semáforo, a tarefa `pmu_tcp_server_out` toma para si o *mutex* de acesso à interface WiFi, monta o frame de dados utilizando o vetor auxiliar da tarefa do SPI e realiza o envio para o PDC. O código-fonte das duas tarefas é apresentado a seguir:

```
void taskSPI( void * pvParameters ){
    configASSERT( ( ( uint32_t ) pvParameters ) == 1 );

    unsigned char checksum; /* Variavel para calculo de checksum */

    /* Variavel utilizada para definir uma transacao SPI */
    spi_slave_transaction_t t;

    memset(&t, 0, sizeof(t));

    while(1){

      memset(recvbuf, 0x0, 129); /* Zera o buffer de recepcao */
```

```
        /* Copia uma mensagem no buffer de transmissao */
        sprintf(sendbuf, (const char* restrict)&phasorResponse);

        /* Define o tamanho maximo dos dados inseridos nos buffers */
        t.length=64*8;

        t.tx_buffer=sendbuf;
        t.rx_buffer=recvbuf;

        /* Linha de handshake setada, para logo apos ser colocada em nivel
           baixo gerando uma borda de descida */
       WRITE_PERI_REG(GPIO_OUT_W1TS_REG, (1<<GPIO_HANDSHAKE));
       vTaskDelay(1);
       WRITE_PERI_REG(GPIO_OUT_W1TC_REG, (1<<GPIO_HANDSHAKE));

        /* Aguarda transmissao por parte do STM32F746G */
        ret=spi_slave_transmit(HSPI_HOST, &t, portMAX_DELAY);

        /* Atualiza o tempo desde a ultima recepcao */
        lastticks = xTaskGetTickCount();

        checksum = 0; /* Zera a variavel de checksum */

        /* Copia os dados recebidos para um vetor auxiliar e calcula o
           checksum */
        for(int i = 0; i <= 42; i++){
             spiData[i] = recvbuf[i];
             checksum += recvbuf[i];
        }

        /* Se o semaforo de transmissao ao PDC ja houver sido instanciado
           (isSyncCreated), se a conexao socket com o PDC ja estiver
           estabelecida e puder ser enviado o frame de dados (data_flag), e
           se o calculo de checksum estiver correto (spiData[43] == checksum)
            entao libere o semaforo, indicando que ha uma transmissao a ser
           realizada */
        if((isSyncCreated == 1) && (data_flag) && spiData[43] == checksum){
             xSemaphoreGive(semSync);
        }
    }

}
```

```
void pmu_tcp_server_out(void *pvParameter){
    configASSERT( ( ( uint32_t ) pvParameters ) == 1 );

    /* Numero de bytes do frame a ser enviado */
    int nbytes;

    /* Instancia o semaforo de transmissao */
    semSync = xSemaphoreCreateBinary();

    isSyncCreated = 1; /*Informa que o semaforo de transmissao foi criado*/
    while(1){
        /* Aguarda a tarefa do SPI liberar o semaforo indicando que ha uma
           transmissao a ser feita */
        xSemaphoreTake(semSync, portMAX_DELAY);

        /* Se houver sido estabelecido uma conexao socket com o PDC */
        if(connected){
            /* Toma o mutex de acesso ao WiFi */
            xSemaphoreTake(mutWIFI, portMAX_DELAY);

            /* Monta o frame de dados e resgata o numero de bytes */
            nbytes = frame_data();

            /* Envia o frame de dados ao PDC */
            lwip_send(newsockfd_out, ucData, nbytes, 0);

            /* Libera o mutex de acesso ao WiFi */
            xSemaphoreGive(mutWIFI);
        }
    }
}
```

5 RESULTADOS

Durante a etapa de migração do código, os periféricos foram sendo configurados um a um, sendo que as primeiras configurações no STM32F746G-*Discovery* foram fazer com que um pulso em um GPIO acionasse um *timer*, que por sua vez disparasse as conversões simultâneas dos três conversores analógicos-digitais. Para que isso seja possível, é necessário que sejam utilizados três conversores A/D separados, e não apenas canais do mesmo conversor. Conforme Figura 23, o *kit* STM32F746G-*Discovery* possui poucos pinos disponíveis para utilização em seus conectores Arduino, e ao verificá-los para definir os conversores AD, constatou-se que haviam dois utilizáveis de maneira prática, e não três. A configuração dos pinos dos conversores analógico-digitais ficou da seguinte maneira:

- ADC1 - Canal 0 - Arduino A0 - Pino PA0 do microcontrolador
- ADC3 - Canal 4 - Arduino A5 - Pino PF6 do microcontrolador
- ADC2 - Canal 4 - Pino da câmera - Pino PA4 do microcontrolador

Devido a isso, foram realizados testes de conversão utilizando os ADCs 1 e 3, sendo o ADC2 deixado com valor flutuando. Os testes com o pulso do GPS realizaram o disparo das conversões com sucesso.

Após configurada a interface UART, ela foi utilizada para receber dados do GPS, que forneceram a estampa de tempo correta. A configuração dos pinos da UART juntamente com o pulso PPS do GPS ficou da seguinte maneira:

- USART6 TX - Arduino D1 - Pino PC6 do microcontrolador
- USART6 RX - Arduino D0 - Pino PC7 do microcontrolador
- *Timer* 1 Canal 1 - Arduino PD10 - Pino PA8 do microcontrolador

Após realizadas essas etapas, foram realizados testes de transmissão de dados fasoriais ao PDC por meio de conexão cabeada com a Internet no *kit* STM32F746G-*Discovery*. Primeiramente, é necessário ativar a conexão com a Internet na tela sensível ao toque, conforme as Figuras 24 e 25.

Depois que o sistema adquire um endereço IP, o PDC pode se conectar a ele para que se inicie a transmissão de dados fasoriais. O *software* utilizado foi o

Figura 24: Ethernet desativada

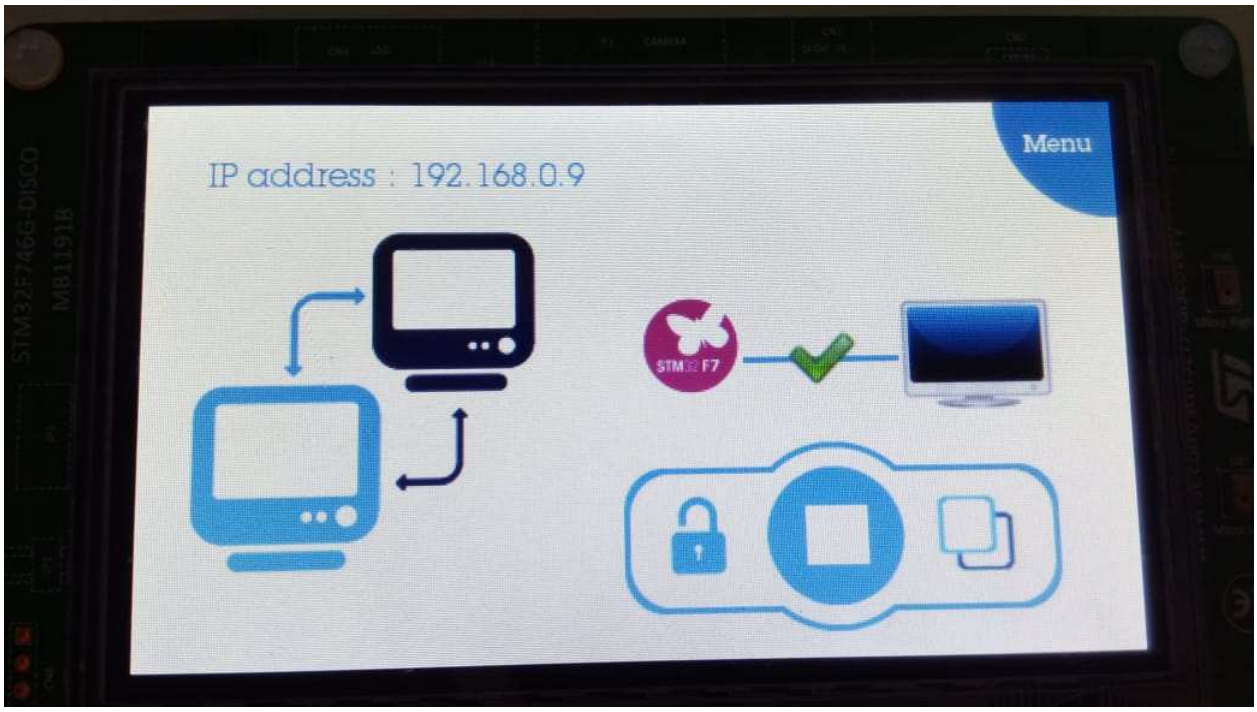

Figura 25: Ethernet ativada

OpenPDC, um concentrador de dados fasoriais *open source* responsável pela recepção de sincrofasores. Ele possui uma plataforma de teste de conexão que é configurada com o IP, porta e código de identificação da PMU. Ao se estabelecer uma conexão, os dados são apresentados na tela e é possível utilizar o arquivo de configuração gerado para finalizar a configuração da PMU no OpenPDC. Foram utilizados

dados de teste para serem transmitidos ao PDC, 30 fasores por segundo. A Figura 26 apresenta essa transmissão, ocorrendo a uma taxa de 0,012 Mbps.

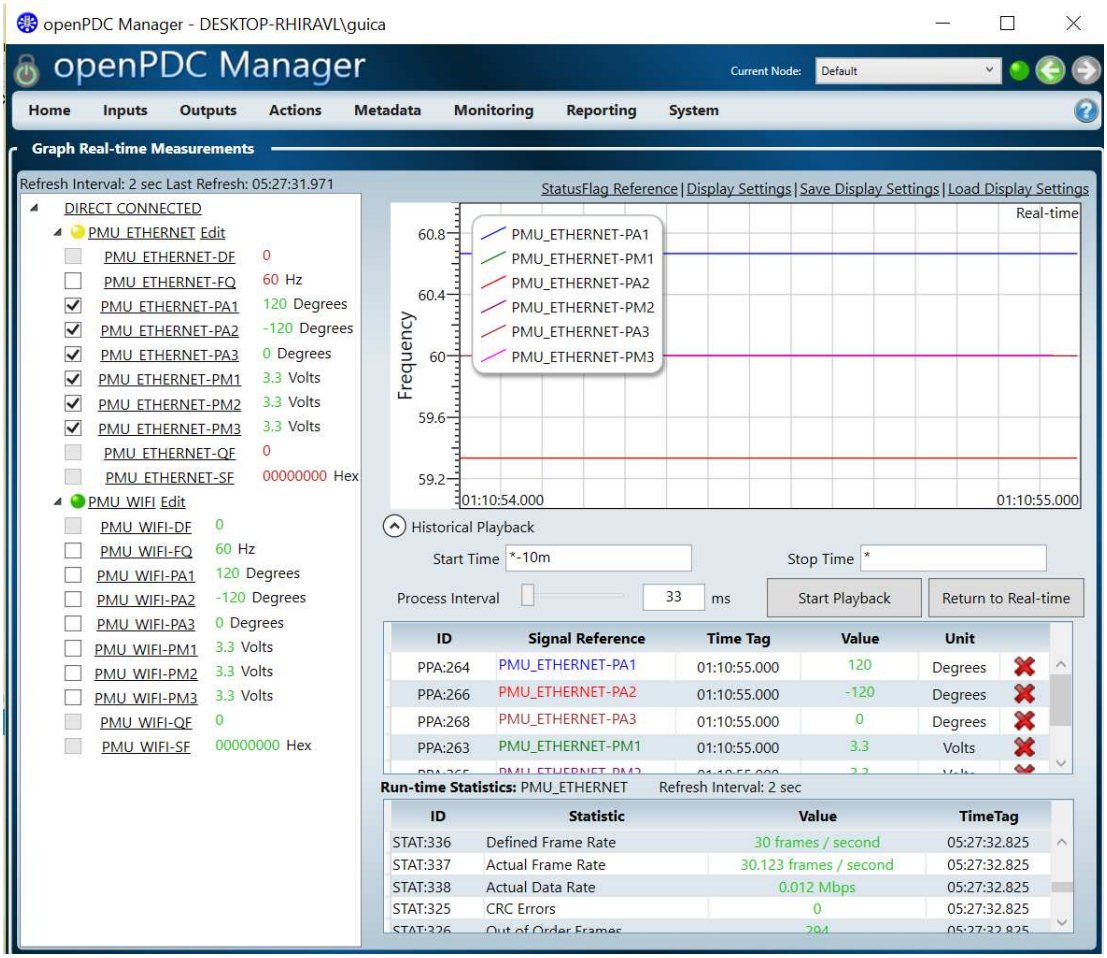

Figura 26: Transmissão de dados fasoriais por conexão cabeada

Após essa etapa, foram realizadas transmissões SPI de dados provenientes do *kit* STM32F746G-*Discovery* para o ESP32, 30 transmissões por segundo. As configurações dos pinos do SPI ficaram da seguinte maneira para o *kit* STM32F746G-*Discovery*:

- MOSI - Arduino D11 - Pino PB15 do microcontrolador
- MISO - Arduino D12 - Pino PB14 do microcontrolador
- SCK - Arduino D13 - Pino PI1 do microcontrolador
- CS - Arduino D5 - Pino PI0 do microcontrolador
- *Handshake* - Arduino D4 - Pino PG7 do microcontrolador

Já as configurações dos pinos do SPI para o ESP32 ficaram da seguinte maneira:

- MOSI - Pino D12
- MISO - Pino D13
- SCK - Pino D15
- CS - Pino D14
- *Handshake* - Pino D2

O *Espressif IoT Development Framework* possui um conjunto de funcionalidades para enviar mensagens para um console na tela, por meio da biblioteca `esp_log.h`. Para isso são utilizadas funções da biblioteca, passando como parâmetro a mensagem que se deseja enviar ao console. Quando essa instrução for executada pelo microcontrolador, é enviada a mensagem informada por meio de conexão serial. Um das utilizações dessa funcionalidade é para realizar depurações, pois o ESP32 não possui *debug* em tempo real. É necessário utilizar esse recurso com cautela, pois o *overhead* associado a ele pode prejudicar a aplicação. Isso foi utilizado para que fosse enviado um texto informando o endereço IP quando este fosse adquirido, além de configurado um LED azul na placa que fica permanentemente ligado quando isso ocorre. Em posse do endereço, é realizada a configuração no OpenPDC da mesma maneira como foi feita por conexão cabeada. A Figura 27 apresenta o envio de dados fasoriais por conexão sem-fio, também a uma taxa de 0,012 Mbps.

Conforme as Figuras 26 e 27 demonstram, a transmissão ocorre de maneira estável nos dois meios de transmissão: cabeado e sem-fio. Além disso, a taxa de dados transmitidos é na ordem de *kilobytes* por segundo. Tanto a transmissão por Ethernet, utilizando conexão cabeada, quanto por WiFi, utilizando conexão sem-fio, possuem taxas de transmissão máximas nominais na ordem de *megabytes* por segundo, sendo então mais do que capazes de efetuarem a transmissão dos dados cumprindo os requisitos de tempo que a aplicação necessita.

Por fim, devido ao *kit* STM32F746G-Discovery e o ESP32 Wroom utilizarem IPs diferentes para efetuarem a transmissão de dados fasoriais ao PDC, é possível realizar o monitoramento de ambos concomitantemente. Desta maneira, a seleção de qual interface monitorar se limita apenas ao *software* OpenPDC, como as Figuras 26

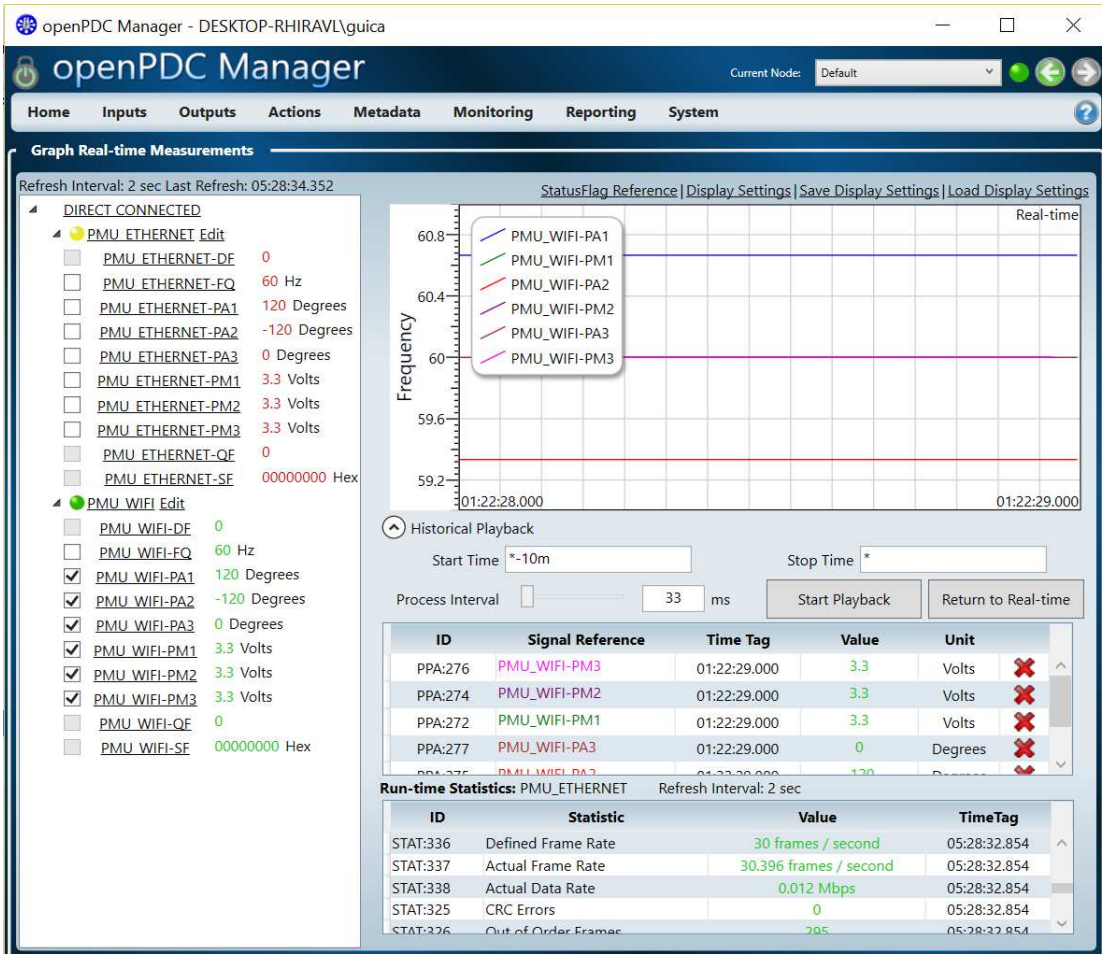

Figura 27: Transmissão de dados fasoriais por conexão sem-fio

e 27 mostram, já que os dois sistemas embarcados estão transmitindo dados fasoriais ao mesmo tempo, cada um em um meio de transmissão diferente.

6 CONCLUSÃO

O trabalho de (GRANDO, 2016) foi desenvolver uma unidade de medição fasorial sincronizada de baixo custo que atendesse aos requisitos de desempenho necessários a nível de distribuição. Seu trabalho foi desenvolvido utilizando o *kit* STM32F429I-*Discovery*, que não possui um módulo Ethernet próprio, utilizando o BRTOS como sistema operacional de tempo real, a *Standard Peripheral Library* como biblioteca de periféricos e o lwIP como pilha de protocolos TCP/IP. Em seu trabalho foi utilizado um módulo Ethernet externo, e o único meio de transmissão de dados fasoriais ao concentrador de dados é por meio de conexão cabeada com a Internet.

Este trabalho se propôs a migrar o código da PMU desenvolvida por Grando (2016) para um *kit* da STM32 com mais recursos, uma tela sensível ao toque maior (o que pode ser utilizado em trabalhos futuros) e um módulo Ethernet próprio, o que torna a transmissão de dados fasoriais mais confiável. Também se propôs a adicionar a funcionalidade de envio desses dados por conexão sem-fio por meio de Wifi. Para isso, foi utilizado a comunicação SPI entre dois microcontroladores.

A migração de código do *kit* STM32F429I-*Discovery* para o *kit* STM32F746G-*Discovery* ocorreu de maneira satisfatória, mantendo as funcionalidades já existentes e sendo atualizado para uma biblioteca de periféricos mais recente como é o STM-Cube Hal, além de utilizar o FreeRTOS com CMSIS como novo sistema operacional de tempo real, tornando futuras migrações de códigos mais simples e mais rápidas de serem realizadas. Além disso, o microprocessador ARM-Cortex M7 no novo *kit* possui maior poder de processamento se comparado ao microprocessador ARM-Cortex M4 no *kit* anterior, possuindo mais que o dobro de DMIPS conforme a Tabela 3 mostra.

No que se refere aos periféricos configurados na migração, um dos conversores analógico-digitais necessários para a estimação fasorial está localizado no pino da câmera, sendo necessário um conector para poder utilizá-lo. Uma opção seria analisar outros *kits* de desenvolvimento da STM32 que sejam da família ARM-Cortex M7 que possuam três conversores AD separados disponíveis para serem acessados nos *headers* Arduino. Agora que a adaptação de *driver* e RTOS foi realizada, uma nova migração de código entre a mesma família de componentes seria mais imediato.

O *kit* de desenvolvimento proposto para adicionar a funcionalidade de envio de dados fasoriais ao PDC por meio de conexão sem-fio foi inicialmente o Simplelink CC3200 da Texas Instruments, mas devido a ele utilizar um outro RTOS com outra pilha de protocolos TCP/IP, optou-se por utilizar o ESP32-Wroom da Espressif Systems por utilizar boa parte dos aspectos de configuração já utilizados na migração do código, como FreeRTOS e pilha de protocolos lwIP. Ao iniciar o desenvolvimento da comunicação SPI entre o *kit* STM32F746G-Discovery e ESP32 Wroom, foram encontrados algumas dificuldades na implementação em relação ao ESP32 devido a falta de depuração em tempo real.

Avançou-se para a próxima etapa e a transmissão de dados por meio de conexão sem-fio funcionou como o esperado, se mantendo tão estável quanto a transmissão cabeada, além de ser possível monitorar ambas concomitantemente, devido a utilização de IPs diferentes em cada sistema embarcado.

Entre as sugestões de trabalhos futuros, há a possibilidade de visualizar o IP do ESP32 Wroom na tela gráfica do *kit* STM32F746G-*Discovery*, e a criação de uma interface gráfica que plote os dados fasoriais diretamente na tela sensível ao toque do *kit*.

REFERÊNCIAS

ALIEXPRESS. **ESP32 DEVELOPMENT BOARD**. 2018. Disponível em:<https://tinyurl.com/ycm7zt2y>. Acesso em: 03 nov. 2018.

BERG, A. M.; SALEHFAR, H.; NEJADPAK, A. Synchrophasor technology: Applications and benefits over conventional measurement. In: **2015 IEEE International Conference on Electro/Information Technology (EIT)**. [S.l.: s.n.], 2015. p. 158–164. ISSN 2154-0357.

BHUIYAN, S. M. A.; KHAN, J. F.; MURPHY, G. V. Big data analysis of the electric power pmu data from smart grid. In: **SoutheastCon 2017**. [S.l.: s.n.], 2017. p. 1–5.

DAVIES, John H. **MSP430 Microcontroller Basics**. 1. ed. [S.l.]: Elsevier, 2008.

ESPRESSIF. **SPI Slave API Reference**. 2018. Disponível em:<https://tinyurl.com/ybmhsgr2>. Acesso em: 03 nov. 2018.

GRAINGER, John J.; STEVENSON, William D. Jr. **Power System Analysis**. 1. ed. [S.l.]: McGraw-Hill Education, 1994.

GRANDO, Flavio L. Arquitetura para o desenvolvimento de unidades de medição fasorial sincronizada no monitoramento a nível de distribuição. Dissertação de Mestrado - Mestrado em Engenharia Elétrica - Programa de Pós-Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná - UTFPR Pato Branco, 2016.

IEEE. Ieee standard for synchrophasor data transfer for power systems. **IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)**, p. 1–53, Dezembro 2011.

IEEE. Ieee standard for synchrophasor measurements for power systems. **IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)**, p. 1–61, Dezembro 2011.

KAEWKIRIYA, T. Performance comparison of wi-fi ieee 802.11ac and wi-fi ieee 802.11n. In: **2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)**. [S.l.: s.n.], 2017. p. 235–240.

KUROSE, James F.; ROSS, Keith W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 6. ed. [S.l.]: Pearson, 2013.

LO, C. H.; ANSARI, N. The progressive smart grid system from both power and communications aspects. **IEEE Communications Surveys Tutorials**, v. 14, n. 3, p. 799–821, Março 2012. ISSN 1553-877X.

NOVIELLO, Carmine. **Mastering STM32**. 0.18. ed. [S.l.]: Leanpub, 2016.

PEREIRA, Fábio. **Microcontroladores MSP430 - Teoria e Prática**. 1. ed. [S.l.]: Érica, 2005.

PHADKE, A. N.; THORP, J. S. **Synchronized Phasor Measurements and Their Applications**. 1. ed. United States of America: Springer US, 2008. (Power Electronics and Power Systems).

SADIKU, Matthew N. O; ALEXANDER, Charles K. **Fundamentos de Circuitos Elétricos**. 5. ed. Porto Alegre: AMGH, 2013.

STMICROELECTRONICS. **Discovery Kit with STM32F746NG MCU**. 2015. Disponível em: <http://www.st.com/resource/en/data_brief/32f746gdiscovery.pdf>. Acesso em: 10 out. 2017.

STMICROELECTRONICS. **Discovery Kit with STM32F429ZI MCU**. 2016. Disponível em: <http://www.st.com/resource/en/data_brief/32f429idiscovery.pdf>. Acesso em: 10 out. 2017.

STMICROELECTRONICS. **STM32F427xx/STM32F429xx Datasheet**. 2016. Disponível em: <http://www.st.com/resource/en/datasheet/stm32f427vg.pdf>. Acesso em: 10 out. 2017.

STMICROELECTRONICS. **STM32F745xx/STM32F746xx Datasheet**. 2016. Disponível em: <http://www.st.com/resource/en/datasheet/stm32f745ie.pdf>. Acesso em: 10 out. 2017.

STMICROELECTRONICS. **STM32 configuration and initialization C code generation**. 2017. Disponível em: <http://www.st.com/resource/en/data_brief/stm32cubemx.pdf>. Acesso em: 10 out. 2017.

TANENBAUM, Andrew S.; WETHERALL, David. **Redes de Computadores**. 5. ed. [S.l.]: Pearson, 2011.

TANENBAUM, Andrew S.; WOODHULL, Albert S. **Operating Systems Design and Implementation**. 3. ed. [S.l.]: Prentice Hall, 2006.