UNIVERSIDADE DO SUL DE SANTA CATARINA

MARTINA KELLI CASETT

DESENVOLVIMENTO DE SOFTWARE DE MONITORAMENTO EM UMA REDE INDUSTRIAL MODBUS

Palhoça

2018

MARTINA KELLI CASETT

DESENVOLVIMENTO DE SOFTWARE DE MONITORAMENTO EM UMA REDE INDUSTRIAL MODBUS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica - Telemática da Universidade do Sul de Santa Catarina como requisito parcial à obtenção do grau de Engenheira Eletricista.

Orientador: Prof. André Tonon, Eng.

Palhoça

2018

MARTINA KELLI CASETT

DESENVOLVIMENTO DE SOFTWARE DE MONITORAMENTO EM UMA REDE INDUSTRIAL MODBUS

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Engenheira Eletricista e aprovado em sua forma final pelo Curso de Engenharia Elétrica - Telemática da Universidade do Sul de Santa Catarina.

Palhoça, 29 de junho de 2018.

Professor e Orientador, André Tonon, Eng.
Universidade do Sul de Santa Catarina

Adriano Mansur Irigoite, Msc./Eng.

Tiago César Busatta, Eng.

Dedico esta, bem como todas as minhas demais conquistas, aos meus pais Roque Valcir Casett e Claudia Regina Andriolli Casett.

AGRADECIMENTOS

Primeiramente a Deus, pelo dom da vida.

Aos meus pais e à minha irmã, que, com muito carinho e apoio, não mediram esforços para que eu chegasse até essa etapa da minha vida.

Agradeço à minha tia e madrinha Salete, por ser uma segunda mãe, a qual ajudou e teve cuidados essenciais.

Ao meu namorado, Bruno, que, com muita paciência e amor, sempre incentivou e aconselhou a seguir caminhos da melhor maneira possível, tornando essa trajetória mais leve e menos complicada.

A todos os professores pelas horas de sabedoria dedicadas a nós, alunos, e, em especial ao professor e orientador, André Tonon, pela sua dedicação, e orientação na conclusão deste trabalho.

Agradeço, também, à empresa Audaces, pelas informações prestadas para que este trabalho se tornasse possível.

A todos, o meu muito obrigada!

“Há uma força motriz mais poderosa que o vapor, a eletricidade e a energia atômica: a vontade.”

(ALBERT EINSTEN)

RESUMO

Redes de comunicação serial são utilizadas em transferências sequenciais de dados através de um único caminho. Entre os protocolos mais empregados nesta forma de comunicação digital, o protocolo MODBUS, devido a sua simplicidade e ampla aplicação, consagra-se como o padrão do mercado. Este trabalho visa à implementação de um canal de comunicações utilizando o protocolo MODBUS no modo de transmissão RTU, adotando o RS-485 como padrão de interface. Esta rede industrial é composta por elementos que integram uma máquina de corte, como inversores de frequência e um analisador de energia, os quais serão objetos de análise no presente estudo. Com o propósito de monitorar os parâmetros obtidos pelos referidos dispositivos de forma integrada e eficiente, será desenvolvido um software em linguagem de programação C# no ambiente Visual Studio.

Palavras-chave: Modbus. Comunicação Serial. Modbus RTU.

ABSTRACT

Serial communication networks are used to sequentially transfer data through a single path. Among the most employed protocols in this method of digital communications, the MODBUS protocol, on account of its simplicity and wide application, is established as the market standard. This work aims at implement a communication channel using the MODBUS protocol in RTU transmission mode, adopting the RS-485 as the interface standard. This industrial network is composed of elements that integrate a cutting machine, such as variable-frequency drives and an energy analyzer, which are going to be the objects of analysis in this work. With the purpose of monitoring the parameters acquired by the referred devices in an integrated and efficient way, a software will be developed using C# programming language in Visual Studio environment.

Keywords: Modbus. Serial Communication. Modbus RTU.

LISTA DE ILUSTRAÇÕES

Figura 1 – Quadro Básico MODBUS. 25
Figura 2 – Quadro MODBUS em rede serial. 26
Figura 3 – Diagrama de estados do Mestre MODBUS. 28
Figura 4 – Diagrama de estados do Escravo MODBUS. 28
Figura 5 – Espaçamento temporal entre caracteres e quadros sucessivos. 34
Figura 6 – Medidor digital de energia Schneider DM6200. 40
Figura 7 – Esquema básico de um conversor de frequência. 43
Figura 8 – Conversor de frequência ABB ACS355 47
Figura 9 – Modelo em cascata 49
Figura 10 – Versão simplificada de um ciclo de vida incremental 49
Figura 11 – Modelo espiral 51
Figura 12 – Modelo de RUP® 52
Figura 13 – Modelo Diagrama de classes 56
Figura 14 – Modelo Diagrama de caso de uso 57
Figura 15 – Modelo Diagrama de sequência 58
Figura 16 – Tela inicial do Visual Studio 2017 61
Figura 17 – Tela para geração de um novo projeto C# 62
Figura 18 – Página inicial do Windows Form Application 63
Figura 19 – Conexões MODBUS – RS485 65
Figura 20 – Configuração da comunicação do medidor de energia 66
Figura 21 – Resultado de dados coletados do Analisador Energia 72
Figura 22 – Conexão MODBUS – RS485 72
Figura 23 – Dados coletados do Inversor de Frequência 75
Figura 24 – Conexão entre dispositivos 76
Figura 25 – Localização do cabo Conversor 77
Figura 26 – Propriedades de Prolific USB-to-Serial Comm Port 77
Figura 27 – Resultado dos dados coletados dos três equipamentos 78
Figura 28 – Interface gráfica para coleta de dados 80

LISTA DE QUADROS

Quadro 1 – Principais diferenças entre redes industriais e convencionais. 19
Quadro 2 – Códigos de Funções MODBUS. 24
Quadro 3 – Comparativo entre padrões de interface serial. 31
Quadro 4 – Comparativo entre protocolos de redes industriais 36
Quadro 5 – Endereço de parâmetro individual 68
Quadro 6 – Função Read Holding Register 70
Quadro 7 – Parâmetro MODBUS 73
Quadro 8 – Configuração da comunicação do Inversor de Frequência 73
Quadro 9 – Parâmetros setados 74

SUMÁRIO

1 INTRODUÇÃO 13
1.1 JUSTIFICATIVA 15
1.2 OBJETIVOS 16
1.2.1 Objetivo geral 16
1.2.2 Objetivos específicos 16
1.3 METODOLOGIA 16
1.4 ORGANIZAÇÃO TEXTUAL 17
2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 PROTOCOLOS DE REDES INDUSTRIAIS 18
2.1.1 Protocolo MODBUS 21
2.1.1.1 Códigos de função 22
2.1.1.2 Quadros MODBUS 24
2.1.1.3 Mecanismo de comunicação MODBUS 26
2.1.1.4 Padrões de interface serial 29
2.1.1.5 Modos de Transmissão MODBUS 31
2.1.1.5.1 MODBUS RTU 32
2.1.1.5.2 MODBUS ASCII 34
2.1.1.5.3 MODBUS TCP 35
2.1.2 Comparativo entre alguns protocolos de redes industriais 36
2.2 DEFINIÇÕES DE POTÊNCIA PARA CIRCUITOS MONOFÁSICOS E TRIFÁSICOS 37
2.2.1 Cálculo de potências elétricas 37
2.3 MEDIDOR DE ENERGIA ELÉTRICA 39
2.3.1 Transformador de corrente 41
2.3.1.1 Classe de exatidão do TC 42
2.3.1.1.1 ABNT 42
2.3.1.1.2 ANSI 42
2.4 INVERSOR DE FREQUÊNCIA 43
2.5 DESENVOLVIMENTO DE SOFTWARE 47
2.5.1 Modelos e metodologias de desenvolvimento de software 48
2.5.1.1 Modelos sequenciais 48
2.5.1.1.1 Modelo clássico - Cascata 48
2.5.1.2 Modelos incrementais 49
2.5.1.2.1 RAD (Rapid Application Development) 50
2.5.1.3 Modelos evolucionários 50

2.5.1.3.1 Modelo de prototipação 50
2.5.1.3.2 Modelo espiral 50
2.5.1.4 Metodologias – Tradicionais ou “pesadas” 51
2.5.1.4.1 Modelo RUP (Processo Unificado da Rational®) 51
2.5.1.5 Metodologias – Ágeis ou “leves” 53
2.5.1.5.1 XP - eXtreme Programming 53
2.5.1.5.2 SCRUM 55
2.5.2 Diagramas UML 55
2.5.2.1 Diagramas estruturais 55
2.5.2.1.1 Diagrama de classes 55
2.5.2.1.2 Diagrama de pacotes 56
2.5.2.1.3 Diagrama de objetos 56
2.5.2.2 Diagramas comportamentais 56
2.5.2.2.1 Diagrama de caso de uso 56
2.5.2.2.2 Diagrama de sequência 57
2.5.2.2.3 Diagrama de atividades 58
2.5.2.2.4 Diagrama de estados 58
2.5.3 Linguagem de programação C# 59
2.5.3.1 Estrutura da linguagem C# 59
2.5.4 Microsoft Visual Studio 60
2.5.4.1 Visual C# 61
2.5.4.2 Windows Form Application 62
3 DESENVOLVIMENTO 64
3.1 ATIVIDADES INICIAIS 64
3.2 CONVERSOR RS485 64
3.3 DESENVOLVIMENTO DO PROGRAMA 65
3.3.1 Biblioteca 65
3.3.2 Comunicação MODBUS com medidor de energia 65
3.3.2.1 Endereçamento 67
3.3.2.2 Código de Função MODBUS 70
3.3.2.2.1 Função 03 – Read Holding Register 70
3.3.2.3 Resultado 71
3.3.3 Comunicação MODBUS com Inversor de Frequência 72
3.3.3.1 Endereçamento 74
3.3.3.2 Código de Função MODBUS 74
3.3.3.3 Resultado 75
3.3.4 Comunicação global MODBUS 76

3.3.4.1 Configuração do Conversor RS485 - USB 76
3.3.5 Interface gráfica.....79
4 CONCLUSÃO 81
REFERÊNCIAS.....82

1 INTRODUÇÃO

Durante muito tempo o Brasil foi dependente da exportação de *commodities*, produtos básicos com pouca ou nenhuma agregação de valor. Para que se pudesse desenvolver a indústria local, no decorrer desse período, o país submetia-se, quase que inteiramente, à importação de máquinas. Nessa época, o setor têxtil de Santa Catarina, que era considerado como um dos mais importantes parques têxteis do país, recorria estritamente à importação de sistemas e equipamentos para automatizar as indústrias de confecção. Todavia, a partir da entrada da Audaces no mercado, a história começou a mudar, principalmente no que se refere ao setor da moda.

No início dessa trajetória, a Audaces se destacou por fornecer ao usuário uma interface amigável, com tecnologias simples de serem utilizadas. Alguns dos principais fatores responsáveis pelo êxito da empresa no seu setor de atuação residiram no fato de as soluções adotadas por seus concorrentes derivarem de outros países e serem embutidas em produtos complexos, que exigiam muito treinamento. Com isso, o suporte e a manutenção desses produtos também eram "importados", resultando em dificuldades financeiras e operativas para os respectivos consumidores.

A Audaces é uma empresa de Florianópolis - SC, fundada em 1992 por Claudio Grando e Ricardo Cunha, dois colegas que se formaram em Ciências da Computação na Universidade Federal de Santa Catarina (UFSC). A empresa desenvolve *softwares* e produz maquinários que proporcionam otimizar a qualidade das peças e a produção das confecções, agregando diversos benefícios aos negócios do setor de vestuário. De forma estratégica, a Audaces começou a exportar em 1996, com foco em países como Argentina e Espanha. Atualmente, a empresa é referência mundial em inovação tecnológica para o mercado da moda e já exporta *softwares* e produtos para mais de 70 países.

Seus produtos são destaques no mercado, pois possuem diferenciais que levam agilidade, qualidade e eficiência aos setores de confecções, móveis, estofados, transportes, entre outros.

Alguns produtos Audaces são elencados a seguir:

- Audaces Vestuário – Primeiro na América Latina – Desde 1997 é o CAD (Computer Aided Design) para modelagem de roupas mais usado por modelistas e encaixadores.
- Audaces Digiflash – Primeiro no mundo – Equipamento que faz a digitalização de moldes de roupas a partir de uma fotografia digital, no mercado desde 2001.
- Audaces JetLux – Primeira na América Latina – Sendo utilizada desde 2004, Jet Lux é uma impressora incomparável de grande formato, usando sua inteligência para plotar riscos perfeitos, gerando economia de tempo e recursos.
- Audaces Idea – Primeiro no Mundo – Desenvolvimento de desenho técnico e ficha técnica para o mercado da moda; através da tecnologia 4DAlize, é possível desenhar sobre um manequim tridimensional, e, em instantes, podem-se materializar as criações em uma impressora 3D.
- Audaces Neocut – Primeiro na América Latina - É a máquina de corte mais segura e inteligente do mundo, atualmente, sendo também fabricada na Itália. Com ela, é possível realizar cortes em vários enfestos e, inclusive, em materiais flexíveis em um mesmo dia. É de fácil operação e utiliza, de forma inteligente, a tecnologia de *IoT (Internet of things)*, cuja premissa fundamental é conectar máquinas à *internet*, fazendo com que essas *smart machines* otimizem a gestão e a produtividade da confecção.
- Audaces 360 – Primeiro no Mundo – Inovação Audaces de 2017, com o intuito de atender todos os tipos de confecções, utiliza um sistema munido de tecnologia 4D, que obtém encaixes com o máximo aproveitamento do tecido. Considerada uma multisolução que propicia multiconectividade.

A demanda contemporânea por processos industriais cada vez mais eficientes e flexíveis faz com que as indústrias visem à otimização de sua cadeia de produção. Para alcançar esse objetivo, tornam-se cada vez mais comuns soluções referentes à inteligência periférica, isto é, à capacidade dos equipamentos, dispositivos e sensores físicos trocarem informações através de uma rede. Um exemplo disso pode ser percebido na modernização de sensores, controladores e

atuadores, responsáveis pela aquisição, monitoramento e correção dos parâmetros de interesse que constituem os processos industriais. Entretanto, frequentes avanços tecnológicos, como os citados, impactam diretamente na diversidade e na abundância de cargas elétricas perturbadoras na rede, tornando o sistema elétrico ainda mais complexo e suscetível a problemas, os quais podem gerar distúrbios no funcionamento de cargas elétricas sensíveis e danificar equipamentos, interrompendo, assim, o processo produtivo das indústrias.

Somando-se a isso as constantes mudanças na legislação que rege o fornecimento de energia elétrica, pode-se compreender por que a qualidade da energia elétrica tem se tornado um fator cada vez mais imprescindível na operação e no desempenho das cargas elétricas industriais. Nesse cenário, a ANEEL – Agência Nacional de Energia Elétrica – foi criada. Essa autarquia especial é responsável por estabelecer normas e padrões mínimos qualitativos, com o objetivo de garantir a comercialização de um produto de qualidade e cujo fornecimento ocorra de maneira otimizada.

Para aumentar a eficiência de um processo industrial que pode ser simplificado como a operação de uma máquina de corte industrial, o presente trabalho propõe-se a desenvolver um *software* para PC que possibilite a interligação de dispositivos distintos (um conjunto formado por 2 *inversores de frequência* e um *analisador de energia*) a uma rede industrial na qual os bits são enquadrados e codificados de acordo com o protocolo MODBUS RTU. A implementação deste sistema de comunicação pretende tornar a troca de informações mais eficiente através da redução de sobrecarga na IHM (interface homem-máquina). Além disso, a implementação do referido *software* facilitará a manutenção e o diagnóstico do sistema, permitindo a leitura dos parâmetros adquiridos e a identificação de falhas, na hipótese de incidência destas.

1.1 JUSTIFICATIVA

Perante o exposto, para aumentar a eficiência de manutenção e diagnóstico dos processos, é necessário que uma comunicação adequada seja estabelecida entre os dispositivos (e.g., sensores, atuadores e controladores) interconectados à rede. Entretanto, quando se carece de compatibilidade, faz-se necessária a busca de soluções alternativas. Consequentemente, para sanar tal

problema, o desenvolvimento de um *software* apresenta-se como uma solução atrativa devido a fatores como, em regra, o baixíssimo custo.

1.2 OBJETIVOS

Para replicar o problema de pesquisa foram estipulados um objetivo geral e dois objetivos específicos.

1.2.1 Objetivo Geral

Desenvolver um *software* que possibilite a integração ótima de dois inversores de frequência, e um medidor de energia que, através de um sistema de comunicação uniforme utilizando o protocolo MODBUS, serão responsáveis pelo controle e monitoramento paramétrico de uma máquina industrial de corte.

1.2.2 Objetivos Específicos

- Realizar testes e avaliar a comunicação MODBUS com os respectivos equipamentos;
- Desenvolver uma interface para coletar dados, falhas e alarmes dos dispositivos.

1.3 METODOLOGIA

- Análise bibliográfica;
- Compreensão do tema;
- Projeto do *software*;
  - Determinar as características estruturais e lógicas da rede de comunicações que será utilizada;
  - Determinar a linguagem de programação e os *softwares* que serão utilizados para o desenvolvimento do aplicativo;
  - Realizar testes e ensaios.

1.4 ORGANIZAÇÃO TEXTUAL

O capítulo 1 tratará dos objetivos gerais e específicos do trabalho, bem como sua relevância e a metodologia utilizada no decorrer de sua execução.

A fim de contextualizar o leitor e facilitar o entendimento deste trabalho, o capítulo 2 apresentará uma revisão bibliográfica sobre os temas que abrangem os assuntos referentes a este trabalho. Como exemplos de assuntos abordados, podem-se citar os seguintes: redes industriais, comunicação serial e seus padrões de interface, protocolo MODBUS, inversores de frequência e analisadores de energia elétrica. Neste capítulo, o aprofundamento bibliográfico sobre cada tema será proporcional à sua respectiva relevância para o trabalho.

O capítulo 3 tratará do desenvolvimento do *software* e do sistema de comunicação integrado através da rede MODBUS. Além disso, serão apresentadas as possíveis dificuldades observadas, as soluções encontradas e os resultados obtidos durante os testes e ensaios.

Por fim, o capítulo 4 irá abordar as considerações finais acerca do desenvolvimento do *software*.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como principais objetivos fornecer as informações necessárias ao leitor para o pleno entendimento do trabalho e construir um direcionamento sobre o tema proposto. O nível de aprofundamento de cada assunto está limitado à sua respectiva importância no contexto em que o trabalho se insere.

Serão abordados assuntos referentes à programação em linguagem C# em um ambiente de desenvolvimento de *software* utilizando o Visual Studio; às redes industriais e sua importância, apresentando os tipos de meios físicos e alguns dos protocolos de comunicação utilizados nesse cenário. Por fim, realizar-se-ão breves relatos e embasamentos acerca dos equipamentos que efetivamente utilizarão o *software* desenvolvido.

2.1 PROTOCOLOS DE REDES INDUSTRIAIS

As redes industriais podem ser descritas como estruturas de comunicação digital que permitem a troca de informações entre os diferentes dispositivos utilizados na geração dos produtos. Essas redes permitem que o tráfego de dados entre os diversos instrumentos e dispositivos que as compõem ocorra através de um único tronco (cabo), dispensando, assim, a necessidade de utilizar cabos distintos para cada dispositivo (AZEVEDO; SOUZA, 2017).

Os processos industriais exigem aquisição adequada e controle preciso das variáveis de interesse (e.g., pressão, nível de um fluido e vazão). Através da comunicação adequada entre tais dispositivos, é possível controlar e manter constantes os parâmetros intrínsecos a uma determinada máquina ou ao processo como um todo. Atingindo-se, assim, um controle mais eficiente do que aquele que seria obtido por um operador (controle manual) no decorrer da produção (BEGA, 2011; ASTRÖM; WITTENMARK, 1997).

O objetivo primário das redes industriais é obter uma comunicação eficiente entre os dispositivos, equipamentos e instrumentos que a constituem, permitindo, dessa forma, a implementação de funções mais avançadas. Algumas das principais vantagens trazidas por essa interação funcional são: aumento global do rendimento do processo; programação, visualização e supervisão remota;

otimização nos processos de aquisição de dados e controle de variáveis (BEGA, 2011; ALBUQUERQUE; ALEXANDRIA, 2009).

As redes industriais podem ser hierarquicamente classificadas de acordo com sua finalidade (i.e., o tipo de aplicação em que operam mais adequadamente). As finalidades, que dependem do tamanho dos pacotes transmitidos na rede, refletem na classificação de diferentes redes industriais, são elas: (a) redes de sensores/atuadores, que transmitem bits em níveis mais baixos; (b) redes intermediárias (*device buses)*, que transmitem quadros de bytes e caracteres; (c) redes elevadas (*field buses*), que são derivadas de instrumentos mais complexos (e.g., controle e instrumentação) e transmitem blocos de informações (BEGA, 2011).

O Quadro 1 mostra as principais diferenças entre redes industriais e convencionais.

Quadro 1 – Principais diferenças entre redes industriais e convencionais.

	Redes Industriais	**Redes Convencionais**
Função primária	Controle de equipamentos físicos	Processamento de dados e transferências
Aplicações comuns	Indústrias/Manufatura	Comércio e residências
Padronização	Diversos protocolos e padrões físicos	Protocolos uniformes
Severidade de falhas	Alta	Baixa
Confiabilidade	Alta	Moderada
Período de *round trip*	250 µs – 10 ms	>= 50 ms
Determinismo	Alto	Baixo

Composição de dados	Pacotes pequenos e tráfego não periódico	Pacotes grandes e não periódicos
Consistência Temporal	Imprescindível	Dispensável
Ambiente Operacional	Condições hostis, geralmente com poeira, calor e vibração	Ambiente “limpo”, especialmente quando há a presença de equipamentos sensíveis

Fonte: Adaptado de GALLOWAY e HANCKE (2012).

As redes industriais proporcionam uma série de vantagens quando comparadas às redes convencionais. Lugli e Santos (2010) citam algumas dessas vantagens: menor quantidade de material (e.g., cabeamento, conectores e canaletas); redução do número de dispositivos; maior flexibilidade para expansões futuras; maior interconectividade e menor mão de obra na instalação e na manutenção.

Em um contexto industrial, os sistemas de controle são ubíquos. Todavia, para que, por exemplo, um controlador possa fazer uso dos dados provenientes de um sensor e, posteriormente, enviar um sinal de controle aos atuadores de um determinado processo, é necessário que a comunicação ocorra de forma eficiente. Diante disso, a padronização e a interoperabilidade dos diferentes protocolos e interfaces existentes são essenciais (ASTRÖM; WITTENMARK, 1997; BEGA, 2011).

As normas referentes à padronização são definidas por diversos órgãos, como o IEEE (*Institute of Electrical and Electronic Engineering*), a ANSI (*American National Standards Institute*) e o INMETRO (Instituto Nacional de Metrologia, Normalização e Qualidade Industrial).

Atualmente, ainda são encontradas diversas soluções de rede proprietárias, isto é, exclusivas dos fornecedores. Isso faz com que os padrões utilizados fiquem restritos aos próprios fabricantes, limitando desenvolvedores e usuários. A padronização tende a otimizar a segurança, a escalabilidade e a integração das tecnologias existentes na rede industrial e, consequentemente, proporciona que o sistema de controle e aquisição de dados opere adequadamente; beneficiando, assim, tanto os desenvolvedores, quanto os usuários finais (LUGLI;

SANTOS, 2011).

Um protocolo, de acordo com Lopez (2000), pode ser conceituado como uma instrução formal de convenções que regem a formatação e a temporização relativa na troca de mensagens entre dois sistemas de comunicação. Ainda, para o autor, um protocolo de comunicação consiste em padrões de *software* e *hardware* responsáveis pela comunicação entre duas estações.

Diferentes protocolos apresentam desempenhos distintos em redes díspares. Devido a isso, pode-se considerar que, no contexto tecnológico contemporâneo, vivemos em um mundo de múltiplos protocolos. Azevedo e Souza (2017) citam alguns dos principais protocolos utilizados nas redes industriais: AS-i, Smart Distributed System, CAN, CANopen MODBUS, Interbus-S, DeviceNet, ControlNet, PROFIBUS e PROFINET.

2.1.1 Protocolo MODBUS

Criado em meados da década de 1970 pela Modicon, inicialmente, o MODBUS era um protocolo proprietário (não aberto) que visava à interconexão de dispositivos, equipamentos, sensores e instrumentos aos controladores lógicos programáveis, que também haviam sido criados pela Modicon. No decurso das décadas seguintes, o MODBUS tornou-se um protocolo aberto e, por conseguinte, o padrão serial adotado pela indústria. Baseado na metodologia pedido-resposta (*request-reply)*, este protocolo transmite informações “brutas” e não apresenta restrições por autenticação ou *overhead* excessivo, o que confere facilidade de implementação, utilização e manutenção ao protocolo. Essa simplicidade[1] é um dos principais fatores responsáveis pelo sucesso do MODBUS e propicia, por exemplo, a comunicação eficiente entre sensores simples (e.g., sensor de temperatura) e computadores sofisticados (CARCANO, 2013; KARNAM, 2007).

Por se tratar de um protocolo da camada de aplicação (embora atue também nas duas camadas mais baixas do modelo OSI/ISO), o MODBUS opera alheio aos protocolos de rede provenientes das camadas mais baixas, podendo, assim, ser utilizado tanto em arquiteturas de redes seriais assíncronas[2], quanto em arquiteturas de redes roteáveis. Diante disso, o MODBUS pode ser implementado

[1] Um indicador disso é a baixa quantidade de páginas (cerca de 100) em sua especificação. A especificação PROFIBUS, por exemplo, contém milhares de páginas.
[2] Em um barramento assíncrono, os dados são enviados sem sincronização de relógio (clock).

em diversos meios físicos: RS-232, RS-422, RS-485, *ethernet*, fibras ópticas, rádio, etc. (GALLOWAY; HANCKE, 2012; KARNAM, 2007).

O MODBUS é um protocolo mestre-escravo[3] (M/S) e seu funcionamento sob condições normais pode ser simplificado da seguinte forma: o mestre controla o acesso e, portanto, a comunicação no barramento; os escravos, que são conectados ao mestre ou a um bloco de *input/output*, são nós[4] que só respondem (transmitindo informações ou realizando operações) quando solicitados pelo mestre. Este, a cada ciclo, pode gravar saídas ou ler entradas em um ou mais (até 247) escravos. O mestre, que é único nó capaz de iniciar transações com outros nós (os escravos não podem dialogar entre si), pode comunicar-se individualmente com cada escravo (*unicast*) ou difundir mensagens para todos os escravos (*broadcast*). Caso os dados requisitados pelo mestre não possam ser interpretados pelo escravo, este envia uma resposta de exceção denominada *exception response* (ALBUQUERQUE; ALEXANDRIA, 2009; MODBUS, 2006; CARCANO, 2013).

A especificação original do protocolo MODBUS estabelece um conjunto de ações de acordo com quatro tipos de formatos de dados: (a) ler ou escrever em bobina; (b) ler entrada discreta; (c) ler registradores de entrada; (d) ler ou escrever em registradores *holding*. No que tange às entradas discretas e às bobinas, é utilizado um único bit. Em contrapartida, operações que envolvam registradores utilizam palavras de 16 bits (CARCANO, 2013; MODBUS, 2006).

No protocolo MODBUS, o endereçamento compreende uma gama de 256 endereços distintos (representados em 8 bits). O endereço 0 é reservado para a comunicação *broadcast*, isto é, quando o mestre envia um pedido a todos os escravos. Cada escravo possui um endereço único, cujo valor deve estar contido na faixa de endereços que vai de 1 até 247. O mestre, por outro lado, não possui um endereço específico (ALBUQUERQUE; ALEXANDRIA, 2009; MODBUS, 2006).

2.1.1.1 Códigos de função

Os códigos de função especificam, basicamente, o tipo de ação que deve ser efetuada pelos escravos mediante requisições do mestre. No quadro MODBUS, esses códigos formam um campo de dados codificado em 8 bits (de 1 a 255, em

[3] Nos modos seriais MODBUS RTU ou MODBUS ASCII.
[4] Um nó, Segundo Bega (2011), é um dispositivo (e.g., mestre, escravo, repetidor ou expansor de I/O) dotado de uma ou mais interfaces de rede.

representação decimal). Entretanto, metade desses endereços (de 128 a 255) é reservada para as respostas de exceção. Com isso, o MSB (*most significant bit*) tem valor 1 para funções de *request* (pedido) e *response* (resposta), e valor 0 para respostas de exceção. De acordo com a especificação da Organização MODBUS (2006), os códigos de funções podem ser classificados em três categorias.

1. **Códigos de Funções Públicas:** são códigos únicos e bem definidos que passaram sob testes de conformidade e validação; englobam tanto códigos previamente designados quanto não designados (reservados para uso futuro).

2. **Códigos de Funções Definidos pelo Usuário:** compreendidos em duas faixas (65 a 72 e 100 a 110), são implementados pelos usuários e tratam de funções que não foram previamente suportadas pela especificação.

3. **Códigos de Funções Reservadas:** indisponíveis para o público geral, visto que são utilizados exclusivamente por uma empresa ou por um conjunto de empresas.

O Quadro 2, a seguir, apresenta os códigos de funções MODBUS.

Quadro 2 – Códigos de Funções MODBUS.

01	0x01	Read Coils	1 bit
02	0x02	Read Discrete Inputs	1 bit
03	0x03	Read Holding Registers	16 bits
04	0x04	Read Input Registers	16 bits
05	0x05	Write Single Coil	1 bit
06	0x06	Write Single Register	16 bits
07	0x07	Read Exception Status	
08	0x08	Diagnostics	
	00 (0x00)	Return Query Data	
	01 (0x01)	Restart Communications Option	
	02 (0x02)	Return Diagnostic Register	
	03 (0x03)	Change ASCII Input Delimiter	
	04 (0x04)	Force Listen Only Mode	
	10 (0x0A)	Clear Counters and Diagnostic Register	
	11 (0x0B)	Return Bus Message Count	
	12 (0x0C)	Return Bus Communication Error Count	
	13 (0x0D)	Return Bus Exception Error Count	
	14 (0x0E)	Return Slave Message Count	
	15 (0x0F)	Return Slave No Response Count	
	16 (0x10)	Return Slave NAK Count	
	17 (0x11)	Return Slave Busy Count	
	18 (0x12)	Return Bus Character Overrun Count	
	19 (0x14)	Clear Overrun Counter and Flag	
11	0x0B	Get Comm Event Counter	
12	0x0C	Get Comm Event Log	
15	0x0F	Write Multiple Coils	1 bits
16	0x10	Write Multiple registers	16 bits
17	0x11	Report Slave ID	
20	0x14	Read File Record	
21	0x15	Write File Record	
22	0x16	Mask Write Register	16 bits
23	0x17	Read/Write Multiple registers	16 bits
24	0x18	Read FIFO Queue	16 bits
43	0x2B	Encapsulated Interface Transport	
	13 (0x0D)	CANopen General Reference Req and Resp PDU	
	14 (0x0E)	Device Identification	

Fonte: CARCANO (2013).

2.1.1.2 Quadros MODBUS

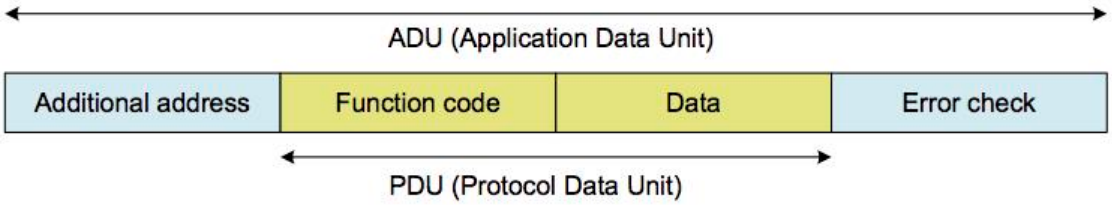
Um quadro, de acordo com Lopez (2000), pode ser definido como um bloco de informações em um protocolo orientado por bits. Conforme Carcano (2013) e Frenzel Junior (2016), as mensagens MODBUS são formadas por unidades de dados conhecidas como ADU (*Application Data Units*). Nos modos seriais (RTU e ASCII) e no modo TCP, uma ADU pode conter no máximo 256 e 260 bytes, respectivamente. Essas unidades são compostas por 2 componentes principais.

1. **PDU (Protocolo Data Unit):** unidades de dados nas quais são inseridas as informações principais. São independentes das camadas inferiores e podem ser do tipo Request PDU (mb_req_pdu), Response PDU (mb_rsp_pdu) ou Exception Response PDU (mb_excep_rsp_pdu).

2. **Campos Adicionais:** são preenchidos pelo mestre e utilizados pelas camadas mais baixas, variam com o tipo de aplicação, mas, essencialmente, destinam-se a viabilizar o mapeamento das mensagens em barramentos ou redes específicas.

A Figura 1 demonstra um *frame* genérico MODBUS.

Figura 1 – Quadro Básico MODBUS.



Fonte: CARCANO (2013).

A Figura 2 apresenta um *frame* MODBUS em uma linha serial (modos RTU ou ASCII). Nesta, os campos apresentados são:

- **Slave ID:** identificador de endereço do escravo, formado por 8 bits (256 níveis: de 0 a 255);

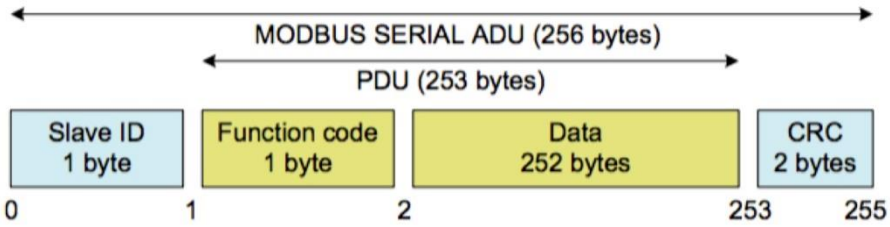
- **Function Code:** é o código da função, que determina o tipo de ação solicitada pelo mestre, formado por 8 bits, resulta em 256 caracteres distintos (de 0 a 255);

- **Data (dados):** contém os parâmetros da pergunta (*request*) e da

resposta (*reply*), os quais são necessários[5] para que a ação solicitada pelo mestre possa ser efetuada (e.g., quantidade de itens a serem manipulados);

- **CRC[6] (Cyclic Redundance Check):** técnica baseada em aritmética polinomial (utilizando um polinômio gerador $g(X) = 1 + X^2 + X^{15} + X^{16}$), é utilizada na detecção de erros durante a transmissão digital de dados. No CRC-16 adotado no MODBUS, para cada *byte* de informação transmitido, o emissor adiciona *bytes* (geralmente 2) de verificação (*checksum*) à mensagem. Esta, ao ser recebida, tem sua integridade verificada pelo receptor que, ao detectar um erro, solicita a retransmissão, enviado um NAK ("n*egative acknowledgement*"). (HAYKIN, 2004)

Figura 2 – Quadro MODBUS em rede serial.



Fonte: CARCANO (2013).

2.1.1.3 Mecanismo de comunicação MODBUS

O MODBUS opera nas camadas 1, 2 e 7 do modelo OSI/ISO. A segunda camada, denominada Camada de Enlace de Dados, é responsável, por exemplo, pelo endereçamento físico e pelo controle de acesso ao meio. No modo de transmissão TCP (Transmission Control Protocol), o mecanismo de acesso ao meio, também denominado "Método de Comunicação" por Azevedo e Souza (2017), é o cliente-servidor. Por outro lado, nos modos de comunicação serial, esse mecanismo de acesso, é conhecido como mestre-escravo (SCHNEIDER ELECTRIC, 2007;

[5] De acordo com a Organização MODBUS (2006), o campo de dados pode ter comprimento nulo em certos tipos de pedidos.
[6] Quando o modo ASCII é utilizado, a técnica de detecção de erros empregada é denominada LRC (Longitudinal Redundancy Check).

AZEVEDO; SOUZA, 2017).

As Figuras 3 e 4 mostram, respectivamente, de forma simplificada e independente do modo de transmissão utilizado, o comportamento do mestre e do escravo durante a transferência de dados no protocolo MODBUS.

Após a energização, no estado inicial, ambos os entes se encontram ociosos. Este estado de ociosidade, aliás, é necessário para evitar que o mestre envie diversos pedidos simultâneos.

Caso se deseje enviar um pedido (*request)* de forma individual a um determinado escravo (*unicast*), o mestre entrará no estado "esperando resposta" e, simultaneamente, com o intuito de evitar que o mestre permaneça neste estado indefinidamente caso não haja resposta, um limitador de tempo (cujo intervalo pode variar conforme a aplicação) conhecido como *response time-out* é ativado. Ao receber uma resposta, o mestre confere se existem erros presentes no quadro e se o quadro recebido foi enviado pelo escravo correto. Caso isto não ocorra, o mestre segue esperando até que receba a resposta do remetente apropriado ou até que o *time-out* expire. Caso isto ocorra, o mestre retornará ao estado ocioso, podendo, então, enviar pedidos a outros escravos ou retransmitir o mesmo pedido para o mesmo escravo (KARNAM, 2007).

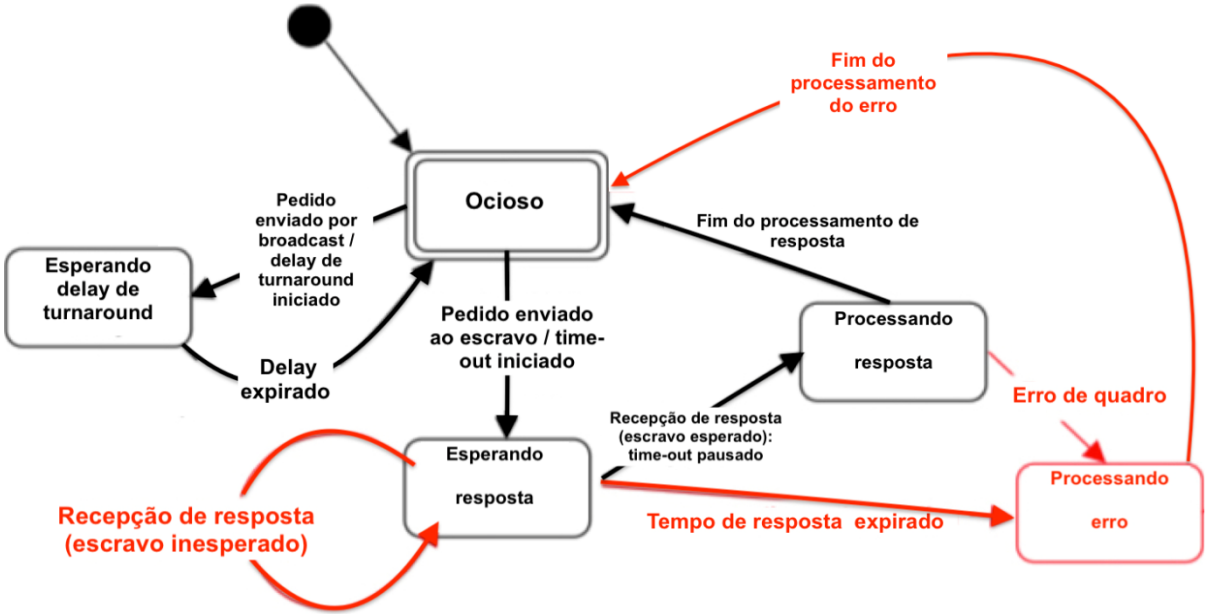
Ao transmitir uma mensagem em *broadcast*, o mestre iniciará outro limitador de tempo designado *delay* de *turnaround* que, para facilitar o processamento (por parte dos escravos) da mensagem difundida, atrasa seu próximo pedido por um determinado período. Quando este período expira, o mestre retorna ao estado "ocioso" sem enviar qualquer resposta ao mestre. Diante disso, no modo *broadcast*, os pedidos enviados pelo mestre são sempre constituídos por comandos de escrita, nunca de leitura.

Ao analisar a comunicação sob a perspectiva do escravo (representada na Figura 3) percebe-se que, ao receber um pacote do mestre, e antes de realizar a ação requisitada, o escravo confere o pacote em busca de erros, como erros de quadro e erros inerentes ao tipo de dado requisitado[7]. Além disso, o escravo confere se ele é o destinatário correto daquele pacote. Em ambos os casos, o escravo retornará ao estado ocioso e aguardará por pedidos posteriores. Por outro lado, na ausência de erros, o escravo processará o pedido requisitado e, em seguida, enviará

[7] Neste caso, conforme indica Karnam (2007), o escravo retornará uma mensagem com o tipo de erro ocorrido através de um código de exceção.

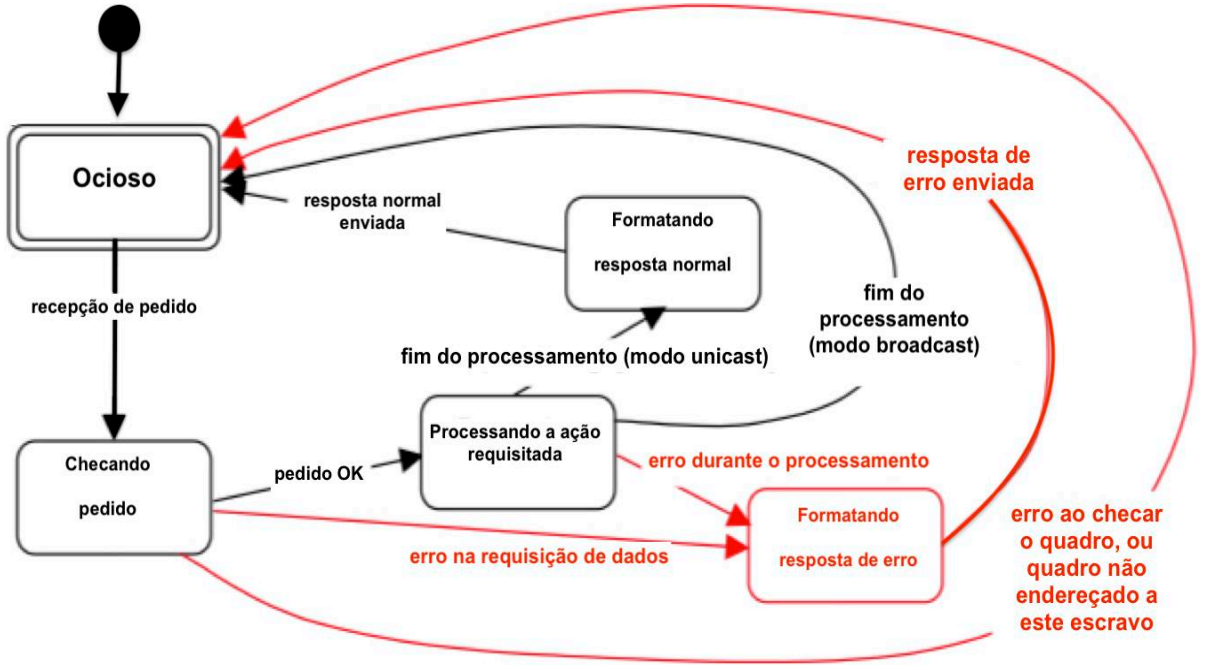
as informações ao mestre (modo *unicast*) ou retornará ao estado ocioso (modo *broadcast*) (CARCANO, 2013; KARNAM, 2007).

Figura 3 – Diagrama de estados do Mestre MODBUS.



Fonte: Adaptado de CARCANO (2013).

Figura 4 – Diagrama de estados do Escravo MODBUS.



Fonte: Adaptado de CARCANO (2013).

2.1.1.4 Padrões de interface serial

Na comunicação digital, a troca de informações entre o(s) transmissor(es) e o(s) receptor(es) pode ocorrer de duas maneiras.

I. **Paralelamente**: por requerer múltiplos caminhos (um para cada bit), a comunicação em paralelo é mais veloz, mas acaba se tornando mais cara e exigindo barramentos e cabos mais complexos. Além disso, é mais suscetível à degradação do sinal devido a fatores como *crosstalk* e interferência eletromagnética.

II. **Em série**: comunicações seriais requerem um único caminho para o tráfego de informações. O fato[8] de exigir uma estrutura física mais simples e barata, aliado a outros fatores (e.g., a utilização obrigatória em comunicações *wireless* ou sistemas ópticos), tornou a transferência serial de dados mais prática e disseminada, resultando em uma vasta gama de interfaces seriais. (FRENZEL JUNIOR, 2016)

Os padrões de interface mais utilizados nas implementações seriais assíncronas MODBUS são (FRENZEL JUNIOR, 2016; LOPEZ, 2000; MODBUS, 2006):

1. **RS-232 (TIA/EIA-232):** É um padrão utilizado em comunicações ponto a ponto (*single ended*) que emprega transmissão *full-duplex* e provê uma interface mínima entre dois dispositivos (DTE e DCE) utilizando apenas três fios em um barramento desbalanceado. Embora seja simples e compatível com uma imensa diversidade de dispositivos, este padrão apresenta limitações significativas no que se refere, por exemplo, à velocidade, à susceptibilidade ao ruído, ao comprimento máximo e à quantidade suportada de nós. Em virtude do exposto, embora já tenha sido o padrão serial mais utilizado, o RS-232 vem sendo gradualmente

[8] Exceção feita aos cabos de fibra óptica.

substituído pelo USB em aplicações comerciais e residenciais (ALBUQUERQUE; ALEXANDRIA, 2009).

2. **RS-422 (TIA/EIA-422):** utilizado em linhas (barramentos) balanceados e diferenciais[9], é capaz de viabilizar a comunicação entre um transmissor e até dez unidades de carga (*load units* ou transceptores). Utiliza dois pares trançados, um para o transmissor (geralmente o mestre) e outro (comutado) para os receptores. A interface RS-422 eleva a taxa de transmissão e a velocidade da RS-232, apresentando um alto nível de imunidade a ruídos. Usualmente, este padrão é utilizado entre apenas dois dispositivos.

3. **RS-485 (TIA/EIA-485):** é o mais difundido e utilizado. Trata-se de uma interface que opera em linhas/barramentos diferenciais e equilibrados, o que aumenta a imunidade ao ruído. Geralmente, é implementado utilizando um par trançado como meio físico de transmissão[10] mas, opcionalmente, caso se deseje um modo de operação *full-duplex*[11] *(simultânea, independente e bidirecional)*, dois pares de cabos podem ser utilizados. O RS-485 é similar ao RS-422, a principal diferença entre eles é que a versão 485 aumenta significativamente a quantidade de conexões, permitindo a comunicação entre até 32 unidades de carga (transceptores). Entretanto, visto que muitos fabricantes produzem transceptores que utilizam apenas uma fração de unidade de carga, o número máximo de transceptores pode exceder a uma centena[12].

O Quadro 3 expõe as principais diferenças entre os três padrões citados.

[9] São linhas nas quais os dispositivos transmitem e recebem dados através da diferença de potencial elétrico entre os condutores do par trançado.
[10] Para Lopez (2000), um meio de transmissão pode ser conceituado como o caminho físico utilizado na propagação dos sinais elétricos e das ondas eletromagnéticas.
[11] Não aumenta a largura de banda da rede.
[12] Podendo chegar a centenas com a utilização de repetidores.

Quadro 3 – Comparativo entre padrões de interface serial.

	RS-232	RS-422	RS-485
Configuração da linha	Single-ended	Diferencial	Diferencial
Modo de operação	Half-duplex Full-duplex	Half-Duplex	Half-Duplex
Topologia	Ponto a ponto	Multidrop	Multiponto
Comprimento máximo	15-30 metros	1200 metros	1200 metros
Quantidade máx. de drivers	1	1	32
Quantidade máxima de receptores	1	10	De 32 a 100
Taxa (máx.) de Transmissão[13]	20k – 115k [bps]	10 M [bps]	50 M[bps]
Sensibilidade dos receptores	±3 [V]	± 200 m [V]	± 200 m [V]
Impedância de entrada mínima	3k – 7 [kΩ]	4 [kΩ]	12 [kΩ]

Fonte: Elaborado pela autora (2018).

2.1.1.5 Modos de Transmissão MODBUS

O termo “MODBUS” faz referência, tipicamente[14], a três variações do mesmo protocolo: MODBUS RTU, MODBUS ASCII e MODBUS TCP. Tais variações, também conhecidas como modos de transmissão, determinam o preenchimento da informação contida nos campos dos quadros, bem como os métodos de transmissão, codificação e decodificação de dados. O modelo de dados e as chamadas de função são similares para os três modos, a principal diferença

[13] A taxa de transmissão reduz à medida que o comprimento da fiação aumenta.
[14] Conforme Albuquerque e Alexandria (2009), outra variação, denominada MODBUS Plus e proprietária da Schneider Electric, também pode ser utilizada sob licença deste fornecedor.

entre eles é o encapsulamento de dados (SCHNEIDER ELECTRIC, 2007; MODBUS, 2006).

Para implementação em comunicações seriais, a especificação MODBUS apresenta duas versões. Ambas empregam o método de comunicação conhecido como mestre-escravo (previamente descrito). A primeira (e mais utilizada) é denominada MODBUS RTU e codifica a mensagem em *bytes* (ou bits). A outra, devido ao fato de utilizar codificação em caracteres ASCII, é chamada de MODBUS ASCII. É fundamental mencionar que, em um barramento serial, o modo de transmissão utilizado deve ser o mesmo para todos os dispositivos. Ademais, todos os dispositivos devem implementar o modo RTU, o ASCII é considerado como um modo de transmissão facultativo (MODBUS, 2006; SCHNEIDER ELECTRIC, 2007).

O MODBUS também permite a implementação em conexões TCP/IP. Este modo, intitulado MODBUS TCP, foi adicionado posteriormente aos modos seriais e pode ser simplificado como um MODBUS RTU que encapsula os pacotes RTU em pacotes TCP/IP (POWELL, 2013; KARNAM, 2007).

2.1.1.5.1 MODBUS RTU

O protocolo MODBUS RTU (*Remote Terminal Unit*) é o protocolo serial mais utilizado por conta, principalmente, da sua densidade de caracteres superior[15]. Tal característica permite um *throughput* mais elevado para a mesma *baud rate*, ou seja, possibilita que o RTU utilize pacotes mais compactos e, devido a isso, apresente melhor desempenho (ALBUQUERQUE; ALEXANDRIA, 2009).

As taxas de sinalização empregadas devem ser de 9,6 kbps ou 19,2 kbps (valor padrão). Na codificação de caracteres RTU, cada *byte* (8 bits) da mensagem é codificado em dois caracteres hexadecimais de quatro bits e, para cada *byte*, são formatados onze bits (um *start bit*, 8 bits de dados, um bit para verificação de paridade e um *stop bit*). Na sequência temporal de bits, o bit menos significativo (LSB) é transmitido primeiro e o MSB é transmitido por último. O comprimento máximo de um quadro serial MODBUS (RTU ou ASCII) é 256 *bytes* (KARNAM, 2007; MODBUS, 2006).

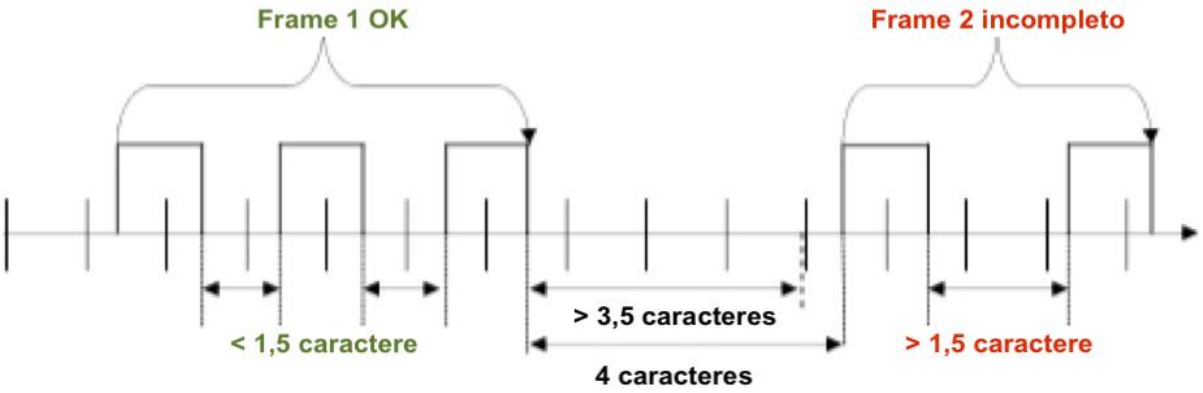
[15] Quando comparado ao MODBUS ASCII.

A verificação de paridade consiste na adição de um bit à mensagem e tem o objetivo de fornecer informações acerca da presença de erros no sinal recebido. No MODBUS, a configuração padrão utiliza paridade par, embora as configurações “paridade ímpar” e “sem paridade” também possam ser empregadas. Nessa última, o bit de paridade é substituído por um *stop bit* extra. Independentemente do modo de verificação de paridade, o RTU utiliza a técnica CRC (previamente descrita) para detectar erros. No CRC, cada quadro enviado é adicionado um campo intitulado “CRC” (ou “*checksum*”), constituído de 2 bytes (16 bits). De forma análoga ao que ocorre com os *bytes* de dados, no campo CRC, o *byte* menos significativo é transmitido antes (LOPEZ, 2000; MODBUS, 2006).

Os quadros RTU não possuem caracteres específicos para indicar os pontos de início e fim de uma mensagem. Portanto, já que no RTU o fluxo de dados deve ocorrer de forma contínua, as sinalizações de início e fim de novos caracteres e quadros são feitas através dos intervalos existente entre caracteres e quadros sucessivos. Tais intervalos, denominados “períodos de silêncio”, permitem, por exemplo, que os escravos detectem e comuniquem erros ao mestre caso ocorra a recepção de mensagens incompletas (MODBUS, 2006).

Os quadros RTU são separados por intervalos iguais ou superiores a 3,5 caracteres. Em contrapartida, os caracteres contidos em um mesmo quadro são temporalmente dispostos em intervalos iguais ou inferiores a 1,5 caracteres. Caso um intervalo superior a 1,5 caracteres e inferior a 3,5 caracteres ocorra entre a transmissão de dois caracteres sucessivos, o quadro é descartado pelo receptor. Na Figura 5, que ilustra esta situação, percebe-se que os períodos de silêncio contidos durante a transmissão do primeiro quadro e entre os dois quadros estão em conformidade com aqueles descritos na especificação. Entretanto, por exceder o limite de tempo (1,5 caracteres $< \Delta t <$ 3,5 caracteres) entre caracteres sucessivos de um mesmo quadro, o segundo *frame* deve ser descartado pelo receptor (MODBUS, 2006; KARNAM, 2007).

Figura 5 – Espaçamento temporal entre caracteres e quadros sucessivos.



Fonte: Adaptado de KARNAM (2007).

2.1.1.5.2 MODBUS ASCII

ASCII (*American Standard Code for Information Interchange*) pode ser definido como um conjunto de códigos de caracteres criado para facilitar a comunicação de dados entre dispositivos de fabricantes distintos. Trata-se de um código binário que representa os dados através de sete bits de informação útil (i.e., informação propriamente dita) acrescentados a um bit de paridade ou símbolos especiais (LOPEZ, 2000).

No modo ASCII (*American Standard Code for Information Interchange*), cada *byte* da mensagem é transmitido e codificado em dois caracteres ASCII. Quando comparada à transmissão no modo RTU, a transmissão em caracteres ASCII aumenta significativamente a quantidade de bits necessários para transmitir a mesma mensagem. Isso reduz a taxa de transmissão do sistema[16], prejudicando a eficiência da comunicação. Por conta disso, embora seja muito similar ao modo RTU e forneça mensagens legíveis (caracteres contidos na tabela ASCII), o protocolo MODBUS ASCII é significativamente menos utilizado em aplicações industriais. Logo, uma explicação mais minuciosa sobre o MODBUS ASCII foge do escopo deste trabalho (MODBUS, 2006; ALBUQUERQUE; ALEXANDRIA, 2009).

[16] Considerando a mesma demanda de recursos (de rede e de processamento) disponível para os dois modos.

2.1.1.5.3 MODBUS TCP

Introduzido em 1999 pela Schneider Electric, o modo TCP é uma variação do protocolo MODBUS que utiliza o conjunto TCP/IP para viabilizar o tráfego de dados utilizando a *ethernet* como meio físico. Neste modo de transmissão, o mecanismo de comunicação é do tipo cliente-servidor (multimestre) e o método de acesso ao meio adotado é o CSMA/CD, próprio da *ethernet* (ALBUQUERQUE; ALEXANDRIA, 2009; SCHNEIDER ELECTRIC, 2007; CARCANO, 2013).

O quadro MODBUS TCP possui 260 bytes e, uma vez que o quadro *ethernet* já emprega o CRC-32, o modo TCP remove os 2 bytes de CRC existentes nos quadros RTU e adiciona um cabeçalho MBAP (*MODBUS Application Protocol*) de 7 bytes. A topologia mais adotada no modo TCP é a “estrela”, embora topologias diferentes (e.g., *daisy chain* e anel) também possam ser implementadas (MODBUS, 2006; SCHNEIDER ELECTRIC, 2007).

Apesar do custo mais elevado de certos equipamentos, maior complexidade e menor compatibilidade com certos tipos de dispositivos (e.g., sensores simples), o modo TCP apresenta certas vantagens em relação aos modos seriais. Algumas delas são: capacidade praticamente ilimitada de dispositivos na rede; taxas de transmissão elevadas (podendo chegar a 1 Gbps); utilização de múltiplos mestre e protocolos de aplicação distintos nos nós (e.g., HTTP) (ALBUQUERQUE; ALEXANDRIA, 2009).

Dessa forma, ao utilizar o protocolo MODBUS, faz-se necessário ponderar sobre a seleção do modo de transmissão utilizado. Essa escolha que, via de regra, limita-se aos modos RTU e TCP, é dependente, principalmente, da aplicação e dos recursos disponíveis. Por exemplo, um sistema que contenha diversos sensores elementares (dotados de baixa capacidade de processamento e memória) não irá usufruir das altas taxas de transmissão fornecidas pelo modo TCP; ainda, nessas aplicações, o uso do MODBUS TCP tende a encarecer consideravelmente o custo do projeto (devido à plataforma necessária para utilizar a *ethernet*) e trazer benefícios apenas ínfimos.

2.1.2 Comparativo entre alguns protocolos de redes industriais

As principais características e parâmetros de alguns dos protocolos industriais são contrastados no Quadro 4.

Quadro 4 – Comparativo entre protocolos de redes industriais.

	MODBUS RTU (RS-485)	PROFIBUS DP/PA	MODBUS TCP	PROFINET	CANopen
Topologia	Linha(*bus*) ou cadeia (*daisy chain)*	Linha, estrela ou anel	Estrela, anel (redundância), linha ou cadeia	Linha, anel, estrela ou árvore	Tronco-derivação
Meio físico	Par trançado	Par trançado ou fibra óptica	Par trançado	Par trançado ou fibra óptica	Par trançado ou fibra óptica
Quantidade (máx.) de nós	32 nós / 246 escravos	32 nós (por segmento)	Praticamente ilimitado (IPV6)	Praticamente ilimitado	127
Distância máxima	1200m	DP: 1200m PA: 1900m Fibra: 24 km	90m (por segmento)	100m (por segmento) ou 26 km (fibra)	≈ 1000m
Método de comunicação	Mestre-escravo	Mestre-escravo	Cliente-servidor	Produtor-consumidor[17]	Produtor-consumidor
Taxa (máx.) de transmissão	10-50 Mbps	DP: 12 Mbps PA: 30 kbps	1 Gbps (fibra óptica)	1 Gbps (fibra óptica)	10-100 Mbps

Fonte: Elaborado pela autora (2018).

[17] Método no qual os dados são classificados como origem ou destino. Apresenta menor tempo de entrega de dados quando comparado ao método Mestre-escravo (AZEVEDO; SOUZA, 2017).

2.2 DEFINIÇÕES DE POTÊNCIA PARA CIRCUITOS MONOFÁSICOS E TRIFÁSICOS

Potência ativa pode ser definida como a parcela da potência complexa capaz de realizar trabalho. Também denominada de Potência Média, a Potência Ativa representa a componente real da Potência Complexa. Por convenção, considera-se que: P > 0 se o elemento está absorvendo energia elétrica (i.e., realizando trabalho ou dissipando energia na forma de calor) e P < 0 se o elemento está fornecendo energia elétrica (e.g., gerador elétrico) (RIZZONI, 2009).

A potência reativa, por outro lado, não absorve energia capaz de realizar trabalho e engloba somente a componente de energia disponibilizada pela fonte que é armazenada em campos elétricos e magnéticos. Essa potência é representada como o componente imaginário da potência complexa. Por convenção, quando uma carga é indutiva, diz-se que sua potência reativa é positiva (Q > 0 [Var]) e a carga está consumindo energia reativa indutiva. Já nos casos de cargas capacitivas (Q < 0 [Var]), considera-se que a carga está fornecendo energia reativa indutiva (ou consumindo energia reativa capacitiva). Por fim, em cargas reativas ideais (para fins teóricos, já que cargas puramente reativas não existem na prática) não ocorre dissipação de energia na forma de calor, isto é, por efeito Joule (UMANS, 2006).

Grande parte dos equipamentos conectados ao sistema elétrico por parte do consumidor, como transformadores e motores elétricos, possuem impedâncias indutivas. Nestas cargas, a tensão aplicada encontra-se adiantada à corrente elétrica (RIZZONI, 2009).

2.2.1 Cálculo de potências elétricas

Para circuitos monofásicos:

Potência Ativa

$$P_a = V_{RMS} \times I_{RMS} \times \cos\emptyset \ \ [W] \tag{1}$$

Potência Aparente

$$S_a = V_{RMS} \times I_{RMS} \ \ [VA] \tag{2}$$

Potência Reativa

$$Q_a = V_{RMS} \times I_{RMS} \times \operatorname{sin} \emptyset \ [VAr] \quad (3)$$

Em que:

$\operatorname{sen} \emptyset = \operatorname{sen}(\emptyset V_{rms} - \emptyset I_{RMS})$

Fator de Potência

$$FP = \frac{\cos \emptyset}{\sqrt{1+TDH^2}} \quad (4)$$

Onde:

TDH = Taxa de Distorção Harmônica

Para circuitos trifásicos equilibrados, isto é, circuitos formados por três cargas de mesma magnitude e fase, temos que:

Potência Ativa

$$P_{3F} = \sqrt{3} \times V_L \times I_L \times \cos \emptyset \ [W] \quad (5)$$

Potência Aparente

$$S_{3F} = \sqrt{3} \times V_L \times I_L \ [VA] \quad (6)$$

Potência Reativa

$$Q_{3F} = \sqrt{3} \times V_L \times I_L \times \operatorname{sin} \emptyset \ [VAr] \quad (7)$$

No que se refere a circuitos trifásico desequilibrados as cargas monofásicas são diferentes em módulo, fase ou ambos, possuindo defasagens distintas entre tensão e corrente. Nesses circuitos, independente da forma de conexão, os valores das potências são obtidos pelas equações que se seguem.

Potência Ativa

$$P_{3F} = P_a + P_b + P_c \ [W] \quad (8)$$

Potência Reativa

$$Q_{3F} = Q_a + Q_b + Q_c \quad [VAr] \tag{9}$$

Potência Aparente

$$S_{3F} = \sqrt{(P_{3F})^2 + (Q_{3F})^2} \quad [VA] \tag{10}$$

Fator de Potência Global

$$FP_{3F} = \frac{P_{3F}}{S_{3F}} \tag{11}$$

2.3 MEDIDOR DE ENERGIA ELÉTRICA

Segundo a Aneel (2017), Qualidade de Energia Elétrica (QEE) pode ser definida como uma medida qualitativa do uso da energia por parte dos consumidores. Essa medida inclui características que englobam a continuidade do suprimento e a conformidade com certos parâmetros, os quais são considerados desejáveis para uma operação segura. A QEE pode ser analisada através de três vertentes principais: qualidade do serviço, qualidade comercial e qualidade do produto. Nessa última, são analisados certos aspectos que dizem respeito, principalmente, à tensão, à frequência e ao fator de potência da energia fornecida pela concessionária.

O monitoramento dos parâmetros relativos à qualidade de energia auxilia no diagnóstico e na análise de eventuais problemas e falhas; proporcionando, assim, diversos benefícios aos usuários. Entre os quais, destacam-se os seguintes: redução de custos, programação (agendamento) de manutenções preventivas e otimização no uso dos equipamentos alimentados pela rede. Nesse contexto, pode-se compreender por que a qualidade da energia elétrica é um fator extremamente importante para os consumidores industriais.

Para monitorar os parâmetros inerentes à QEE, o sistema a que este trabalho se refere utiliza um medidor de energia digital Easylogic DM6200, fabricado pela Schneider Electric. Este medidor (Figura 6) agrega uma porta RS-485 à versão básica (DM6000) da série, possibilitando a transferência de dados através do protocolo MODBUS RTU.

Algumas das principais características do DM6200 são descritas a seguir (SCHNEIDER ELECTRIC, 2015):

1. Instalação e configuração simples e rápidas.
2. Possui placas que contêm códigos de cores para otimizar a fiação.
3. Possibilita proteger a visualização dos parâmetros por meio de senha.
4. Visor inteligente: provê acesso veloz aos parâmetros prioritários, exibe barra de carga[18] (com códigos de cores), contém rolagem automática e autoajuste da ordem de grandeza (e.g., Quilo para Mega) dos valores.
5. Conexão direta para tensões de linha de até 480 [Vca].
6. Medição de Tensão (incluindo desequilíbrio e ângulo de fase), frequência, potência, fator de potência, quantidade de interrupções, horas de funcionamento e velocidade de rotação em [rpm][19].

Figura 6 – Medidor digital de energia Schneider Electric DM6200.

Fonte: SCHNEIDER ELETRIC. Disponível em: <https://www.schneider-electric.com.ph/en/product/METSEDM6200/dm6200-digital-meter-with-basic-readings---with-communication>. Acesso em: 11 jun. 2018.

[18] Dotada de 12 segmentos LEDS, informa o consumo atual da instalação (em Amperes %): cada segmento LED representa 10% do valor nominal.
[19] Apenas para geradores (SCHNEIDER ELECTRIC, 2015).

O DM6200 é empregado no modo MODBUS RTU através do padrão de interface elétrica RS-485. Neste modo, adota-se a topologia *daisy chain* (também conhecida como “cadeia” ou “varal”). Além do medidor DM6200, podem ser conectados até 31 dispositivos de 2 fios, totalizando 32 dispositivos na rede MODBUS. O medidor utiliza o endereço 1 (decimal) e a verificação de paridade utilizada é a “verificação par” (MODBUS, 2006; SCHNEIDER ELECTRIC, 2015).

De acordo com a Schneider Electric (2015), a leitura dos dados desses dispositivos é feita pelo código de função 03 (*Read Holding Registers*) e a taxa de baud[20] que pode ser alcançada[21] nesta forma de comunicação é 19200 [bauds].

2.3.1 Transformador de corrente

Um transformador de corrente (TC) é um dispositivo que, através de um acoplamento magnético, reproduz, em um enrolamento secundário, uma réplica em magnitude (sem alterar a fase) da corrente que circula no enrolamento primário (BIRD, 2009; PETRUZELLA, 2014).

Para KINDERMANN (2012), esse tipo de transformador de instrumentação possui três finalidades essenciais:

- Prover isolação elétrica entre os dispositivos conectados no secundário (equipamentos de medição, controle e relés) e o circuito de alta tensão (primário);
- Fornecer uma corrente no secundário que seja proporcional à do primário;
- Fornecer no secundário uma corrente com dimensões apropriadas aos equipamentos de medição e proteção.

Um dos parâmetros mais importantes para a especificação de um TC é a relação de transformação (RTC), a qual pode ser definida como a relação entre as correntes nominais do primário e secundário. Por exemplo, uma RTC de 75/5

[20] Baud Rate = R [bps] / n, em que “n” é a quantidade de bits utilizados para representar um símbolo, nível ou caractere.
[21] Para distâncias inferiores a 300 metros e considerando um canal sujeito a degradações (e.g., ruído, atenuação e interferências) típicas.

informa que, para uma corrente de 75 [A] no primário, circulará uma corrente de 5 [A] no secundário (SANCHES, 2010).

Segundo Kindermann (2012), a carga no secundário do TC é a máxima carga que pode se conectar do secundário do transformador sem ultrapassar a máxima tensão especificada pela sua classe de exatidão e de acordo com a equação:

$$\dot{V}_{2,máximo} = \sum \dot{Z}_{carga} * \dot{I}_2 \ [V] \quad (12)$$

2.3.1.1 Classe de exatidão do TC

A classe de exatidão (CE) pode ser conceituada como o limite garantido de erro em qualquer medição efetuada. Transformadores de corrente utilizados com a finalidade de medição (baixas classes de exatidão resultam em leituras mais precisas) comumente saturam para uma corrente de $4\ \mathrm{I}_{\mathrm{nominal}}$. Logo, na ocorrência de um curto-circuito com magnitudes superiores ao quádruplo da corrente nominal do respectivo enrolamento, o secundário do TC excederá o erro de sua CE devido à saturação do núcleo magnético. Assim, correntes secundárias maiores do que 20 [A] aumentam significativamente os erros de leitura, gerando medidas imprecisas (KINDERMANN, 2012).

2.3.1.1.1 ABNT

Para a ABNT, a classe de exatidão do TC é a máxima potência aparente consumida pela carga conectada no secundário, para que a corrente nominal no secundário seja de 5A (KINDERMANN, 2012).

2.3.1.1.2 ANSI

A classe de exatidão do TC, conforme a ANSI, é a tensão máxima que pode ser induzida no secundário durante a máxima corrente de curto-circuito, em função de seu fator de sobrecorrente. Em TCs de medição, como visto, o fator de sobrecorrente é tipicamente igual a 4. Por outro lado, no caso dos TCs de proteção, geralmente empregados em aplicações que envolvem correntes de falta mais

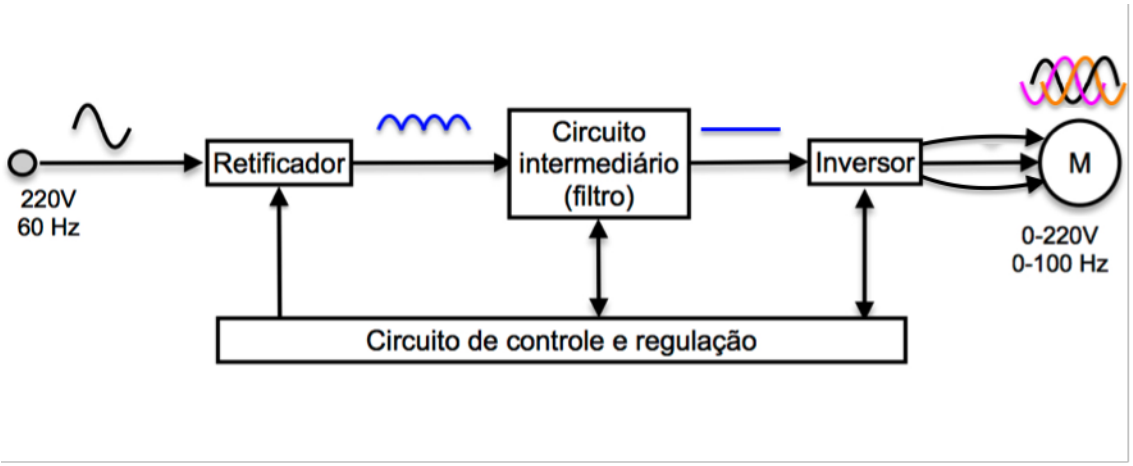
severas, o fator de sobrecorrente é geralmente igual a 20 (KINDERMANN, 2012).

2.4 INVERSOR DE FREQUÊNCIA

Em um ambiente industrial, os motores elétricos são responsáveis por uma parcela considerável do consumo de energia elétrica. Para reduzir esse custo e permitir uma operação adequada em aplicações que exigem velocidade variável, o controle da velocidade do motor é uma técnica amplamente utilizada e eficiente. Há alguns anos isso era feito, majoritariamente, pela aplicação de motores de corrente contínua. Entretanto, apesar de oferecerem um controle preciso, os motores CC costumam apresentar problemas, como a necessidade de manutenção frequente (devido às escovas) e o faiscamento. Assim, e diante da evolução tecnológica na eletrônica de potência, foram desenvolvidos diversos equipamentos que otimizaram as aplicações nas quais o controle de velocidade era utilizado. Entre os dispositivos desenvolvidos utilizando componentes microeletrônicos, pode-se destacar o inversor de frequência, também denominado de conversor de frequência (CHAPMAN, 2013; CREDER, 2016).

Conforme Rizzoni (2009), um inversor de frequência é formado por quatro subsistemas principais, ilustrados na Figura 7.

Figura 7 – Esquema básico de um conversor de frequência.



Fonte: Elaborado pela autora (2018), com base no esquema apresentado por RIZZONI (2009).

Conforme mostra a Figura 7, a tensão elétrica CA (60 Hz) da rede, por intermédio de um retificador em onda completa, é transformada em tensão contínua.

Em seguida, esta tensão CC é submetida à ação de um filtro (provido de um capacitor, que reduz o *ripple* de tensão), tem seu valor alterado para aproximadamente:

$$V_{cc} = \sqrt{2} \times V_{ca,rede}\ [V] \tag{13}$$

Posteriormente, a tensão de saída do filtro é novamente transformada em uma tensão CA com magnitude e frequência ajustáveis. Isso ocorre pela ação de dispositivos como os IGBTs (Transistores Bipolares de Porta Isolada), que são chaveados por meio de um modulador PWM (*Pulse Width Modulation*) senoidal. Este, ao utilizar uma frequência de modulação em Hertz [Hz], possibilita que uma corrente aproximadamente senoidal flua pelo enrolamento de armadura do motor CA. Embora, idealmente (considerando a existência nula ou insignificante de componentes harmônicos na tensão e na corrente), a tensão aplicada aos terminais do motor, seja estritamente sinusoidal, isso não ocorre na prática (FRANCHI, 2010; RIZZONI, 2009).

O circuito de controle e regulação, que dispõe de um modulador PWM, é responsável por ajustar a tensão e a frequência do sinal de saída. Tais valores independem da tensão da rede elétrica. Dessa forma, é possível obter uma tensão de saída com frequência superior àquela da rede elétrica. O controle do chaveamento é feito de duas maneiras (FRANCHI, 2010; CREDER, 2016).

1. **Controle Escalar**: utilizado em aplicações que dispensam o controle do conjugado do motor. De acordo com Franchi (2010), este método consiste em controlar a tensão e a frequência da máquina através de uma alimentação em malha aberta (sem realimentação). Visto que a variação do torque é proporcional à relação entre a variação de tensão e a variação de frequência, o torque, que é proporcional ao fluxo magnético, permanece constante para qualquer valor de velocidade. Isto é:

$$\Delta C \propto \Delta\varphi = \frac{\Delta V}{\Delta f} = cte.$$

2. **Controle Vetorial**: os dispositivos munidos deste método de controle dinâmico geralmente apresentam um controle em malha fechada e são equipados de um *encoder*, também apresenta o modo vetorial *sensorless*. Adotando a corrente do motor como referência, o método vetorial possibilita controlar, de forma precisa[22] e simultânea, a velocidade e o torque do motor. Além de um alto desempenho dinâmico, o controle vetorial apresenta oscilações suaves no intervalo de velocidades especificado pelo fornecedor e é tipicamente empregado em aplicações que requerem extrema precisão de velocidade ou conjugado elevado na partida (e.g., elevadores).

Os inversores de frequência permitem acionar, sem aquecimentos exagerados ou vibrações excessivas, cargas de torque constante ou variável, reduzindo suas correntes de partida através da partida em rampa (suave). Além disso, permitem reduzir de forma significativa o consumo de energia elétrica, e operam com alto fator de potência e rendimento superior a 90% para toda a faixa de velocidades (cuja variação pode chegar até 1:20). Diante do exposto, essas máquinas apresentam inúmeras aplicações industriais (e.g., em bombas hidráulicas, para reduzir o golpe de aríete[23]), sendo usualmente empregados no acionamento de motores de indução do tipo gaiola de esquilo com frequências entre 1 e 60 [Hz].

Franchi (2010) representa o inversor de frequência em quatro blocos distintos:

1. **Unidade Central de Processamento (CPU):** formada por um microprocessador associado a uma memória ou por um microcontrolador, a CPU é incumbida de armazenar os parâmetros e dados do sistema, bem como de fornecer os pulsos de disparo aos IGBTs.

2. **Interface Homem-Máquina (IHM)**: é um dispositivo de I/O que possibilita, entre outras coisas, alterar os parâmetros do motor (e.g.,

[22] Principalmente no que tange à velocidade, visto que o conjugado eletromagnético do motor pode apresentar pequenas oscilações com à variação da carga mecânica.
[23] Conforme Fox, Pritchard e McDonald (2011), o golpe de aríete é causado devido às reflexões presentes na propagação de ondas acústicas em um liquido confinado. Tais reflexões podem ser causadas, por exemplo, pelo fechamento abrupto de uma válvula.

velocidade de rotação e tempo de aceleração), ligar ou desligar o inversor e, através do *display*, visualizar as grandezas de interesse.

3. **Interfaces:** propiciam o comando do inversor através de sinais analógicos e digitais.

   a. Analógicas: estes sinais, cujos módulos tipicamente estão situados entre 0 e 10 [Vcc] (0-20 m[A] ou 4-20 m[A]) e são proporcionais à velocidade de rotação do motor, permitem controlar a velocidade e inverter o sentido de rotação do motor. Para isto, basta trocar a polaridade do sinal analógico. Um exemplo de interface analógica tipicamente encontrado nos inversores de frequência é o padrão serial RS-485, que permite controlar e monitorar o motor de forma remota.

   b. Digitais: os sinais discretos são responsáveis, por exemplo, pelo controle e monitoramento do conversor, isso ocorre através de funções tipicamente simples. Uma dessas funções é o controle de velocidade conhecido como *multispeed,* que permite escolher até oito velocidades de rotação pré-programadas.

4. **Etapa de Potência**: contém o circuito retificador, que é encarregado de fornecer a tensão contínua ao módulo IGBT através do barramento CC.

Apesar das diversas vantagens referentes ao emprego dos inversores de frequência para controlar a velocidade dos motores elétricos, há que se considerarem algumas condições operativas que podem deteriorar seu desempenho. Uma dessas condições, tanto para Franchi (2010), quanto para Mamede Filho (2015), é a distorção harmônica. Esse fenômeno, oriundo da tensão aplicada aos terminais do motor, que não apresenta um comportamento plenamente senoidal, introduz componentes harmônicos indesejáveis. Estes, elevam as perdas térmicas e a temperatura do motor, prejudicando sua eficiência. Para manter a temperatura do motor em conformidade com sua classe de isolamento, recomenda-se multiplicar o conjugado do motor por um fator de redução. Este fator, que depende da taxa de distorção harmônica contida no sistema, vai de 0,73 até 1,00.

A máquina de corte Neocut Bravo - Audaces utiliza quatro conversores de frequência, modelo ACS355, fabricados pela empresa ABB.

Figura 8 – Conversor de frequência ABB ACS355

Fonte: ABB. Disponível em: <https://search-ext.abb.com/library/Download.aspx?DocumentID=3AUA0000081917&LanguageCode=en&DocumentPartId=&Action=Launch>. Acesso em: 12 jun. 2018.

2.5 DESENVOLVIMENTO DE SOFTWARE

Em meados dos anos 60/70, com a “Crise do Software”, a Engenharia de Software surgiu para realizar projetos de desenvolvimento de *software* de uma maneira mais organizada. Dessa forma, possibilitando a geração de processos estruturados, planejados e padronizados, para assim poder atender às necessidades dos usuários finais com mais eficiência, dentro de um cronograma e orçamento previsíveis (PRESSMAN, 2006).

Para que esses processos ocorressem de forma adequada, surgiram os diagramas UML (*Unified Modeling Language*), e os modelos e metodologias de desenvolvimento. Estes dividem o desenvolvimento de *software* em procedimentos pré-estabelecidos, modelando e facilitando o entendimento do produto (SOMMERVILLE, 2000).

2.5.1 Modelos e metodologias de desenvolvimento de software

2.5.1.1 Modelos sequenciais

São modelos que organizam as fases do processo em ordem linear. O principal modelo sequencial é o modelo Clássico - Cascata, porém, pode-se citar também o modelo em V, o qual é menos utilizado (PRESSMAN; MAXIM, 2012).

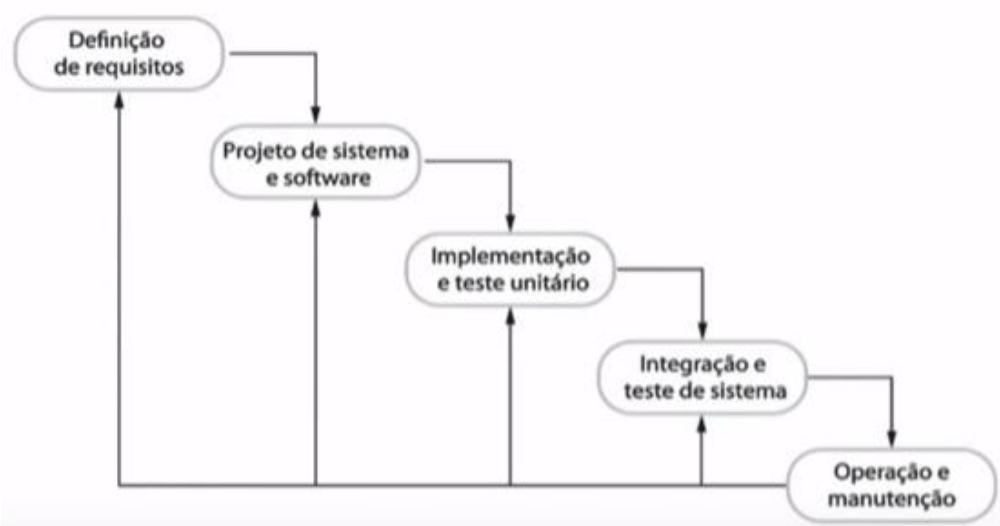
2.5.1.1.1 Modelo clássico - Cascata

É um modelo em que a fase seguinte só pode ser executada quando a fase anterior estiver totalmente finalizada (PRESSMAN, 2006).

O modelo é iniciado pela definição de requisitos, nesta fase são definidas as funções, restrições e objetivos do sistema. Após tais definições, procede-se para a fase de projetos de sistemas e de *software*, na qual são agrupados os requisitos de sistemas de *hardware* e *software*. Em seguida, passa-se para a fase de implementação e teste de unidades, na qual verifica se as especificações de cada unidade foram atendidas.

Após a conclusão da fase de implementação, segue-se para a fase de integração e testes de sistemas. Nesta etapa, as unidades de programas são integradas e testadas. Depois da verificação do teste de sistemas, o *software* é entregue ao cliente. Por fim, a última fase, denominada “Operação e Manutenção”, trata da instalação do sistema e da correção de erros em todas as fases envolvidas (PRESSMAN, 2006; SOMMERVILLE, 2003).

Figura 9 – Modelo em cascata

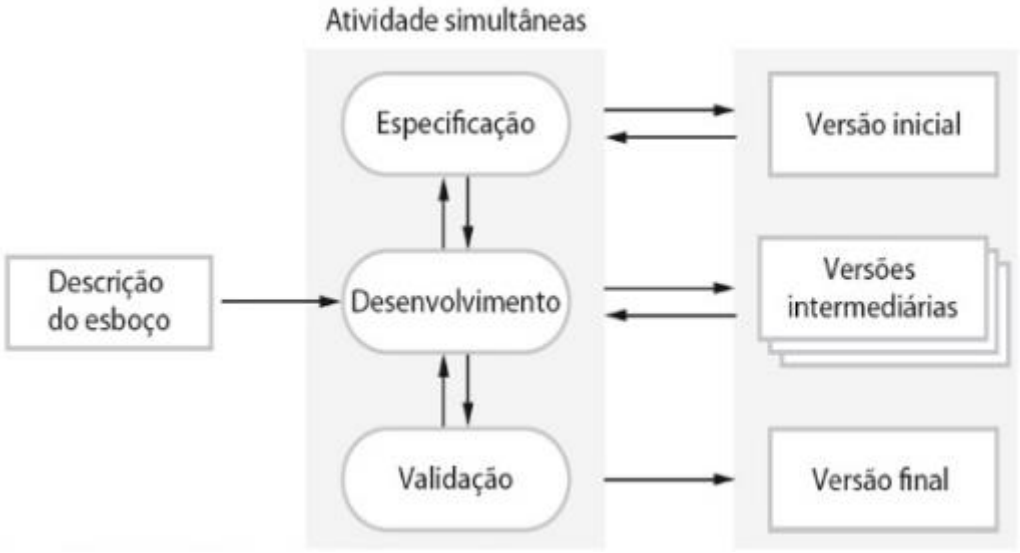


Fonte: SOMMERVILLE (2011).

2.5.1.2 Modelos incrementais

O modelo incremental, cujo ciclo de vida simplificado é apresentado na Figura 10, tem o propósito de apresentar um novo produto operacional em sucessivos incrementos (PRESSMAN, 2006).

Figura 10 – Versão simplificada de um ciclo de vida incremental



Fonte: SOMMERVILLE (2011).

2.5.1.2.1 RAD (Rapid Application Development)

Desenvolvimento rápido de aplicação (RAD) é um modelo de processo de curto desenvolvimento, recomendado para projetos onde as principais funcionalidades são implementadas em menos de três meses. Neste modelo o sistema também é separado em subsistemas, gerando incrementos. A diferença fundamental entre o modelo incremental e o RAD é que, nesse último, os incrementos são desenvolvidos em paralelo e por equipes distintas, sendo entregue apenas a versão final do sistema (PRESSMAN, 2006).

2.5.1.3 Modelos evolucionários

Segundo Pressman (2006), modelos evolucionários produzem sistemas que atendem de imediato às demandas de clientes. A criação desses modelos serviu, entre outras coisas, para que *softwares* sejam entregues em versões reduzidas, ou seja, com requisitos básicos, que, com o passar do tempo, podem ser implementados por outras versões com funcionalidades mais complexas, a fim de completar o projeto inicial.

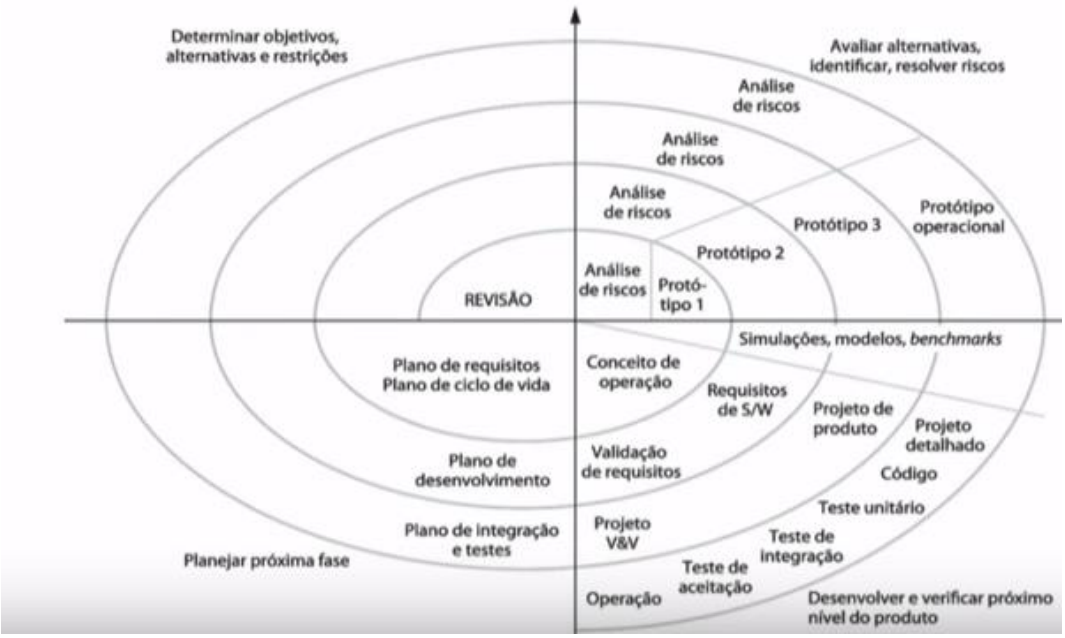
2.5.1.3.1 Modelo de prototipação

A prototipagem tem como principal característica a geração de protótipos do sistema, com as definições de requisitos dadas pelo cliente. O teste desse protótipo é feito pelo cliente e visa à validação das funcionalidades (PRESSMAN, 2006).

2.5.1.3.2 Modelo espiral

O modelo espiral é um modelo complexo em técnicas de desenho. A filosofia desse modelo é unir incrementos e iterações, a fim de evitar problemas futuros. Neste tipo de modelo, definem-se objetivos, alternativas e restrições do sistema, analisam-se riscos e, por fim, fazem-se protótipos. Suas atividades ficam mais complexas à medida que evolui em sua forma espiral (SOMMERVILLE, 2011).

Figura 11 – Modelo espiral



Fonte: SOMMERVILLE (2011).

2.5.1.4 Metodologias – Tradicionais ou “pesadas”

As metodologias tradicionais ou “pesadas” geralmente são desenvolvidas no modelo “cascata”, porém, para alteração de documentação, é necessário retornar ao início do processo. São metodologias consideradas lentas e burocráticas, pois se aplicam a projetos vultosos e envolvem grandes equipes de desenvolvimento, tornando a conclusão do processo excessivamente longa (PRESSMAN, 2006).

2.5.1.4.1 Modelo RUP (Processo Unificado da Rational®)

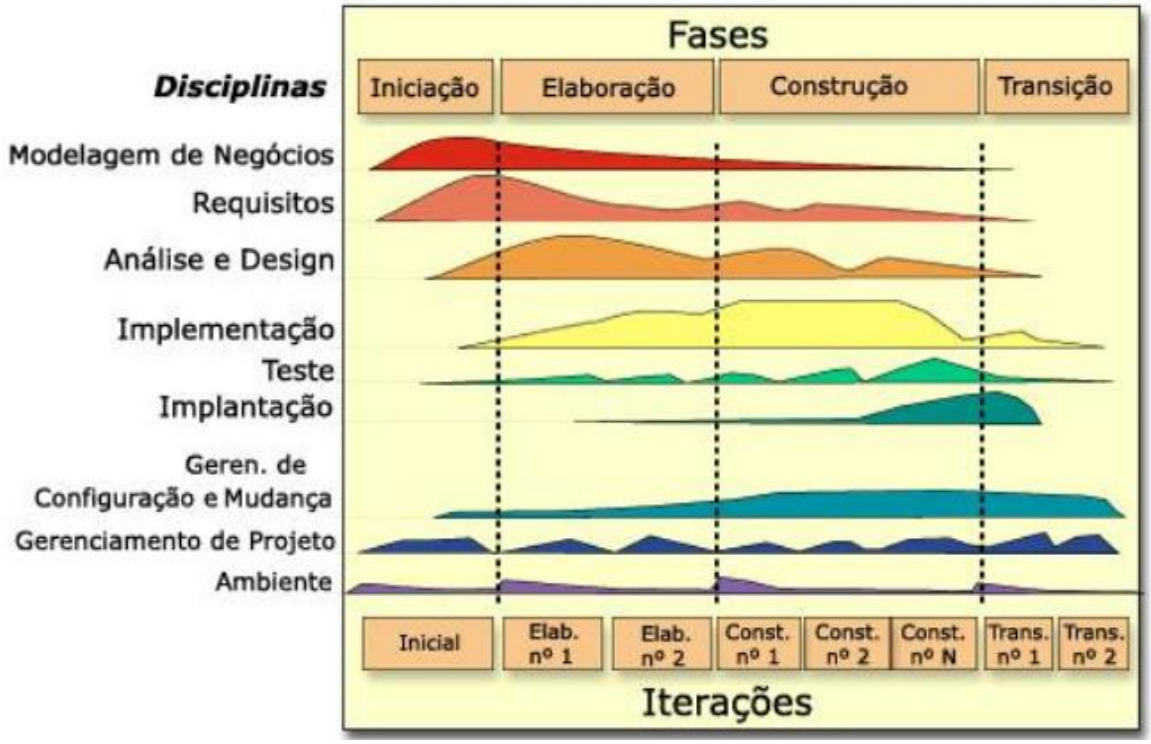
Criado pela Rational Software Corporation e posteriormente adquirido pela IBM, o RUP é um processo proprietário da Engenharia de Software. Este modelo foi criado para auxiliar o desenvolvimento de aplicações orientado a objetos, utilizando a Linguagem de Modelagem Unificada (UML).

Pode-se dizer que o modelo RUP é um modelo de processo híbrido, pois forma um conjunto dos modelos de prototipação, incrementais e iterativos; relacionando três perspectivas diferentes: dinâmica (fases), estática (disciplinas) e prática (iteração) (IBM, 2001).

As melhores práticas referentes ao RUP segundo Tamaki (2007) são:

- Desenvolvimento de *software* iterativo;
- modelagem visual;
- gerenciamento de requisitos;
- arquitetura baseada em componentes;
- gerenciamento de mudanças;
- verificação contínua de qualidade.

Figura 12 – Modelo de RUP®



Fonte: IBM (2001).

Devido ao desenvolvimento iterativo, o RUP permite que a entrega de *software* seja realizada om mais frequência e agilidade, permitindo que a equipe de desenvolvimento tenha uma quantidade maior de *feedback* do cliente para, assim, adequar o *software* para atender de forma ótima às expectativas dos clientes (OSORIO; CHAUDRON; HEIJSTEK, 2011).

Todavia, tal processo é inadequado para projetos de pequeno porte. Isso ocorre em consequência da sua alta complexidade e necessidade de uma equipe de desenvolvimento com vasta experiência na manipulação

de ferramentas e métodos.

2.5.1.5 Metodologias – Ágeis ou “leves”

Como o próprio nome indica, os métodos ágeis servem para acelerar o ritmo de processos de desenvolvimento de *software*, reduzindo o ciclo de desenvolvimento em semanas ou meses.

Os métodos ágeis caracterizam-se pelo caráter adaptativo e orientado a pessoas. Segundo o Manifesto Ágil (2001), alguns princípios que definem essa metodologia são elencados a seguir:

- Entregas rápidas e frequentes, com menor escala de tempo possível;
- Customização de produtos;
- O funcionamento do *software* é a medida primária de progresso;
- Equipes de desenvolvimento e negócios trabalham juntas e são, via de regra, formadas por poucas pessoas;
- Processos ágeis promovem o desenvolvimento sustentável;
- Organização, simplicidade e agilidade nos projetos.

O termo “agilidade” é muito visado no contexto das metodologias ágeis, pois há a necessidade de responder de forma rápida e adequada a certas modificações no decorrer do processo. Com isso, o processo deve ser projetado de modo que permita à equipe de projeto adaptar e aperfeiçoar tarefas, focando nos produtos de trabalho mais essenciais e mantendo-os simples.

A seguir, serão brevemente explicados os métodos mais conhecidos dessa metodologia.

2.5.1.5.1 XP - eXtreme Programming

A metodologia ágil XP (eXtreme Programming) é considerada uma abordagem deliberada e disciplinada, que provê agilidade no desenvolvimento de soluções, economia de recursos e qualidade do produto final.

Segundo Beck (2004), a XP possui cinco valores:

- **Comunicação:** é de extrema importância que haja interação entre o cliente e a equipe de desenvolvimento, já que muitos problemas carecem de solução devido à falta de comunicação.
- **Simplicidade:** criação de *softwares* fáceis e viáveis.
- **Feedback:** para um bom desenvolvimento, é necessário o *feedback* em cada fase do projeto.
- **Coragem:** para corrigir e remover possíveis erros e, até mesmo, remover uma parte grande e importante do código para, então, refazer um novo projeto de arquitetura.
- **Respeito:** Respeitar e se interessar no trabalho da equipe pois, sem essa relação, o método XP torna-se ineficaz.

Além dos valores, também é necessário levar em consideração as práticas de trabalho, as quais garantem a efetividade do trabalho da equipe XP e a satisfação do cliente durante o desenvolvimento do processo.

As práticas mais utilizadas para uma melhor eficiência em projetos de *software* são (BECK, 2004):

- Jogo de Planejamento;
- fases pequenas;
- metáforas;
- projeto simples;
- testes constantes;
- *refactoring;*
- programação em pares;
- padronização do código;
- propriedade coletiva;
- integração contínua;
- semana de 40 horas;
- cliente no local.

2.5.1.5.2 SCRUM

O método SCRUM é um *framework* utilizado para gerenciar o desenvolvimento de produtos complexos, ou seja, pode ser integrado a outros métodos ágeis com facilidade. Aplica-se tanto ao desenvolvimento de *software*, quanto a qualquer ambiente de trabalho. O SCRUM é baseado em equipes pequenas e com ciclos que dividem o desenvolvimento em intervalos de tempos de, no máximo, 30 dias. Tais ciclos, chamados de *sprints*, são atividades de desenvolvimento que seguem um padrão de processo, são elas: requisitos, análises, projeto, evolução e entrega (SCHWABER; SUTHERLAND, 2011).

2.5.2 Diagramas UML

Diagramas UML (*Unified Modelling Language)* não são considerados métodos de desenvolvimento, são tipos de linguagens em forma de diagramas que auxiliam os desenvolvedores a especificar, visualizar, construir e documentar modelos de *softwares* orientados por objetos (BOOCH; RUMBAUGH; JACOBSON, 2005).

Os diagramas UML são divididos em Diagramas Estruturais (estáticos) e Diagramas Comportamentais (dinâmicos). Na sequência, serão destacados os diagramas UML mais usualmente utilizados (BOOCH; RUMBAUGH; JACOBSON, 2005).

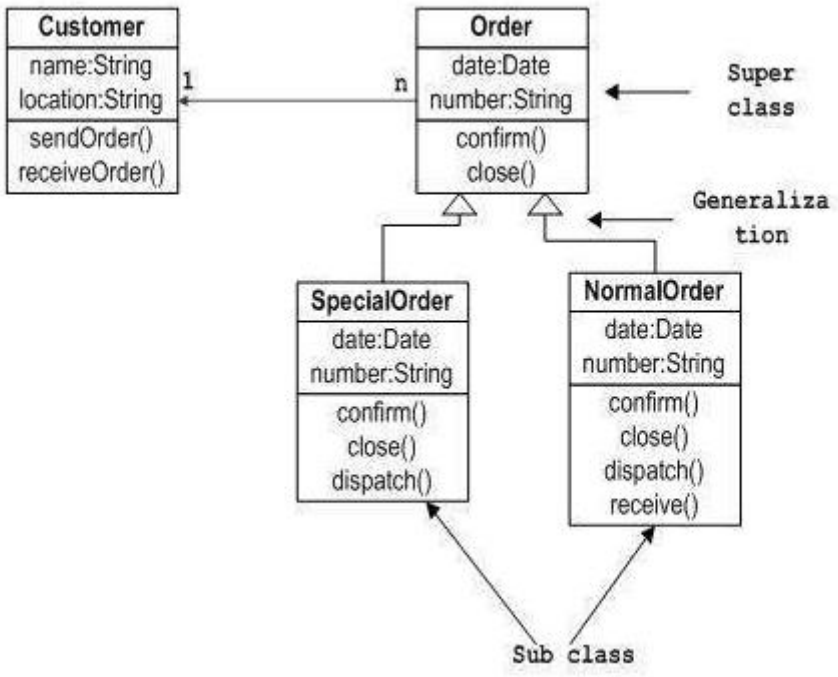
2.5.2.1 Diagramas estruturais

Os principais diagramas estruturais são: Diagrama de classes, Diagrama de Pacotes, Diagrama de Pacotes e Diagrama de Objetos (BOOCH; RUMBAUGH; JACOBSON, 2005).

2.5.2.1.1 Diagrama de classes

O diagrama de classes define a estrutura de classes do sistema, sendo considerado o mais importante para a UML, já que serve de apoio para a maioria dos outros diagramas (BOOCH; RUMBAUGH; JACOBSON, 2005).

Figura 13 – Modelo Diagrama de classes



Fonte: Disponível em: <https://www.tutorialspoint.com/uml/uml_class_diagram.htm>. Acesso em: 12 jun. 2018.

2.5.2.1.2 Diagrama de pacotes

O diagrama de pacotes tem como principal objetivo transformar as classes em pacotes (BOOCH; RUMBAUGH; JACOBSON, 2005).

2.5.2.1.3 Diagrama de objetos

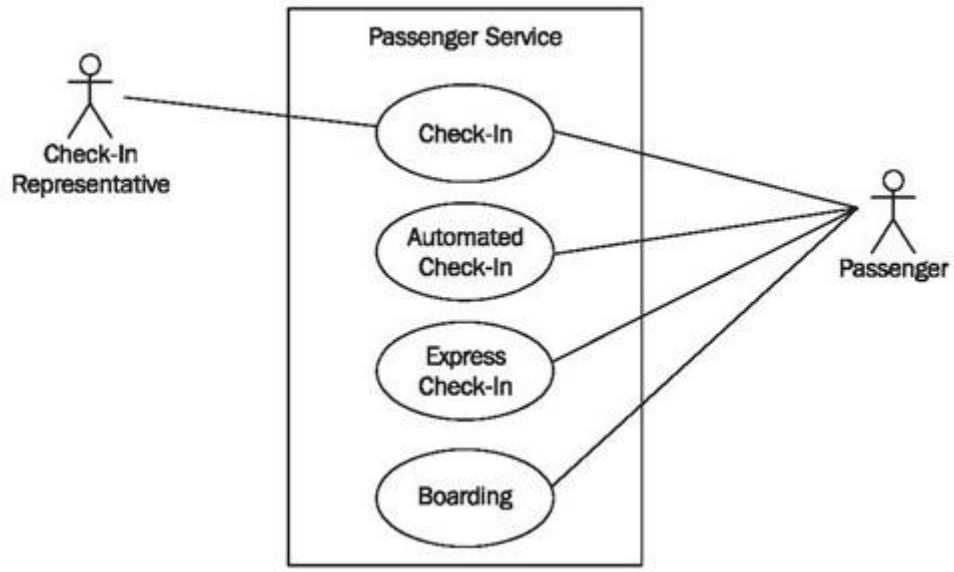
Pode-se considerar o diagrama de objetos como um complemento do diagrama de classes, uma vez que representa os valores armazenados pelos objetos de um diagrama de classes em um determinado instante de tempo (BOOCH; RUMBAUGH; JACOBSON, 2005).

2.5.2.2 Diagramas comportamentais

2.5.2.2.1 Diagrama de caso de uso

Diagrama de caso de uso é o diagrama mais simples da UML e mostra as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários daquele sistema. Para que, assim, possam ser definidos os passos necessários para atingir a um objetivo específico (PRESSMAN; MAXIM, 2012).

Figura 14 – Modelo Diagrama de caso de uso



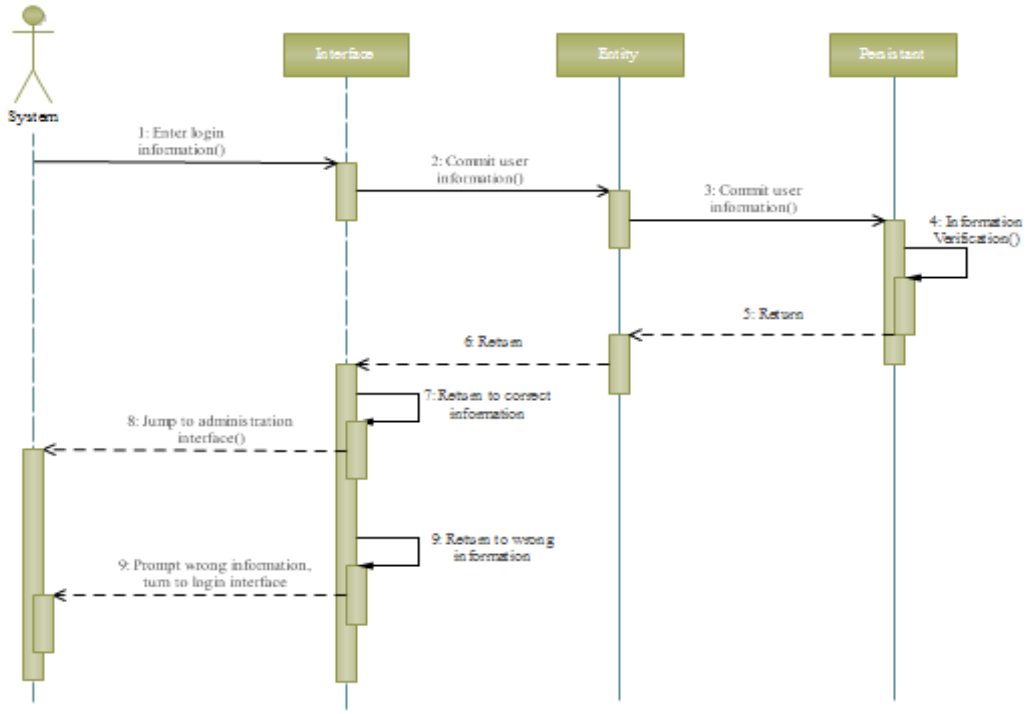
Fonte: <https://sourcemaking.com/uml/modeling-business-systems/external-view/constructing-use-case-diagrams>. Acesso em: 12 jun. 2018.

2.5.2.2.2 Diagrama de sequência

É utilizado para indicar as comunicações dinâmicas entre objetos durante a execução de uma tarefa. Além disso, enfatiza a ordem temporal na qual as mensagens são enviadas entre os objetos para executar aquela tarefa, podendo se basear em um caso de uso (PRESSMAN; MAXIM, 2012).

Um diagrama de sequência mostra as chamadas de método utilizando setas horizontais do “chamador” para o “chamado”, identificadas com o nome do método, incluindo seus parâmetros, seus tipos e o tipo de retorno.

Figura 15 – Modelo Diagrama de sequência



Fonte: Disponível em: <https://www.edrawsoft.com>. Acesso em: 12 jun. 2018.

2.5.2.2.3 Diagrama de atividades

O diagrama de atividades mostra um comportamento dinâmico de um determinado sistema e descreve as atividades que devem ser executadas através da representação do fluxo de controle de um processo (BOOCH; RUMBAUGH; JACOBSON, 2005; PRESSMAN; MAXIM, 2012).

O fluxo de controle é indicado por setas que vão de um “nó ação” para outro. “Nó ação” é o componente principal de um diagrama de atividades, o qual corresponde a uma tarefa executada por um sistema de *software* e cuja representação se dá por um retângulo (PRESSMAN; MAXIM, 2012).

2.5.2.2.4 Diagrama de estados

Este diagrama mostra uma máquina de estados e consiste em estados, transições, eventos e atividades. Geralmente, os diagramas de estado são utilizados para ilustrar a visão dinâmica de um sistema (BOOCH; RUMBAUGH; JACOBSON, 2005).

2.5.3 Linguagem de programação C#

A linguagem de programação C# (C Sharp), além de segura, é bem simples de se implementar. Isso torna o C# fácil de ser aprendido e utilizado, robusto e com bom desempenho. Esta é uma linguagem orientada a objetos fortemente tipada. Desenvolvido pela Microsoft, o C# visa ao desenvolvimento de uma variedade de aplicações que são executadas sobre a plataforma .NET.

A plataforma .NET é uma plataforma de *software* relacionada a uma vasta variedade de tecnologias, conectando informações, sistemas, pessoas e dispositivos. A linguagem C# foi criada para ser a principal linguagem de programação desta plataforma (MICROSOFT, 2018).

O C# é a simplificação da linguagem C/C++ no que se refere a áreas de classes, *namespaces*, gerenciamento de exceções e sobrecarga de métodos. O C Sharp propõe-se a tornar o desenvolvimento e a utilização mais fáceis e simples, diminuindo a taxa de erro de programação (WILLE, 2001).

2.5.3.1 Estrutura da linguagem C#

C# é constituído em uma estrutura que possui uma classe externa, o nome do projeto, o nome da classe e o procedimento principal. Conforme Fukuyoshi (2012), as funções de cada componente dessa estrutura são listadas a seguir:

1. **Classes externas:** classes básicas necessárias ao funcionamento adequado do programa, determinam pacotes que serão utilizados por ele.

2. **Nome do projeto:** nome para o projeto que será iniciado, declara um escopo que contém um conjunto de classes relacionadas.

3. **Nome da classe:** especifica o nome da classe que está sendo criada, podendo ser alterado a qualquer momento.

4. **Procedimento principal:** lugar onde é inserido o código.

Exemplo dos segmentos da estrutura:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Exemplo01
{

    class Program
  {

      static void Main(string[] args)
    {


    }

  }

}
```

Classes externas

Nome do projeto

Nome da classe

Procedimento principal

2.5.4 Microsoft Visual Studio

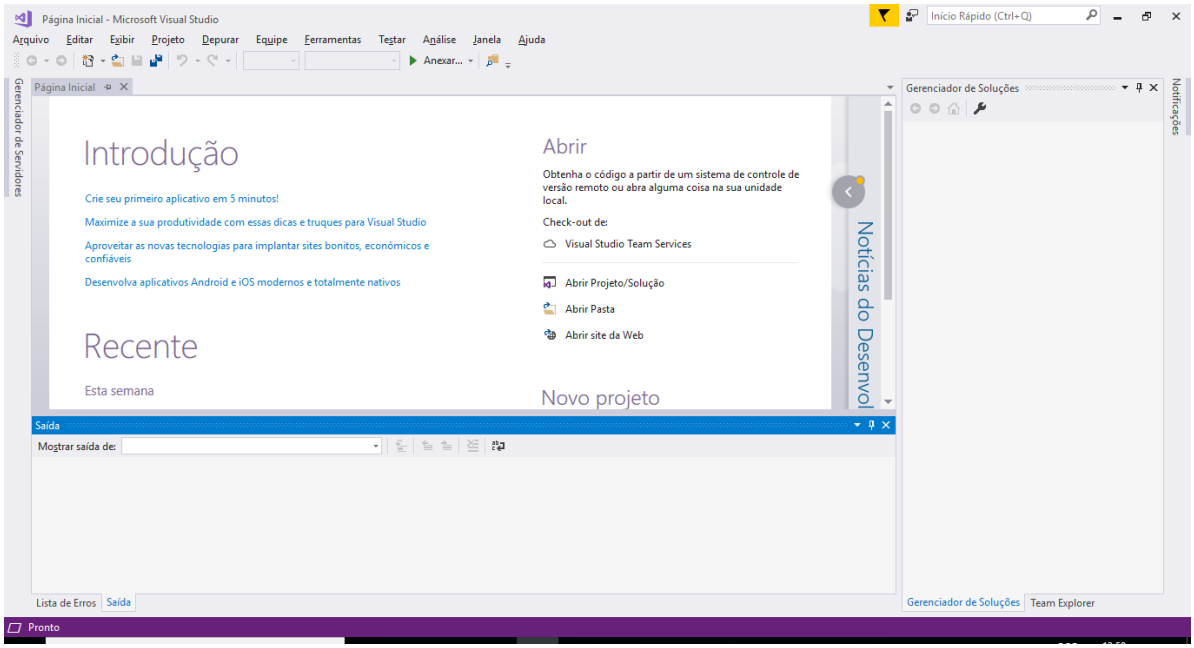
Microsoft Visual Studio é um ambiente de desenvolvimento interativo (IDE) de linguagem de programas como C#, Visual Basic, Visual C#, Visual F#, Visual C++ e JavaScript, e oferece um conjunto de ferramentas que auxiliarão na elaboração e edição do código, bem como na detecção e correção de erros. (MICROSOFT, 2015).

A ferramenta de desenvolvimento Visual Studio é robusta e completa; sua proposta é apresentar ao usuário o que está sendo produzido, podendo ser utilizada para editar e mostrar quase todo tipo de código, compilando e publicando aplicativos para Web, Android, IOS, Windows, etc. Nessa perspectiva, tornam-se essenciais as

ferramentas de *debug* e análise do código, que possibilitam avaliar a performance da aplicação e os erros.

No Visual Studio 2017, como demonstra a Figura 16, é possível visualizar os recentes projetos utilizados, abrir um novo projeto, visualizar ícones de criação de novos projetos, ler dicas referentes ao uso do *software*, etc.

Figura 16 – Tela inicial do Visual Studio 2017



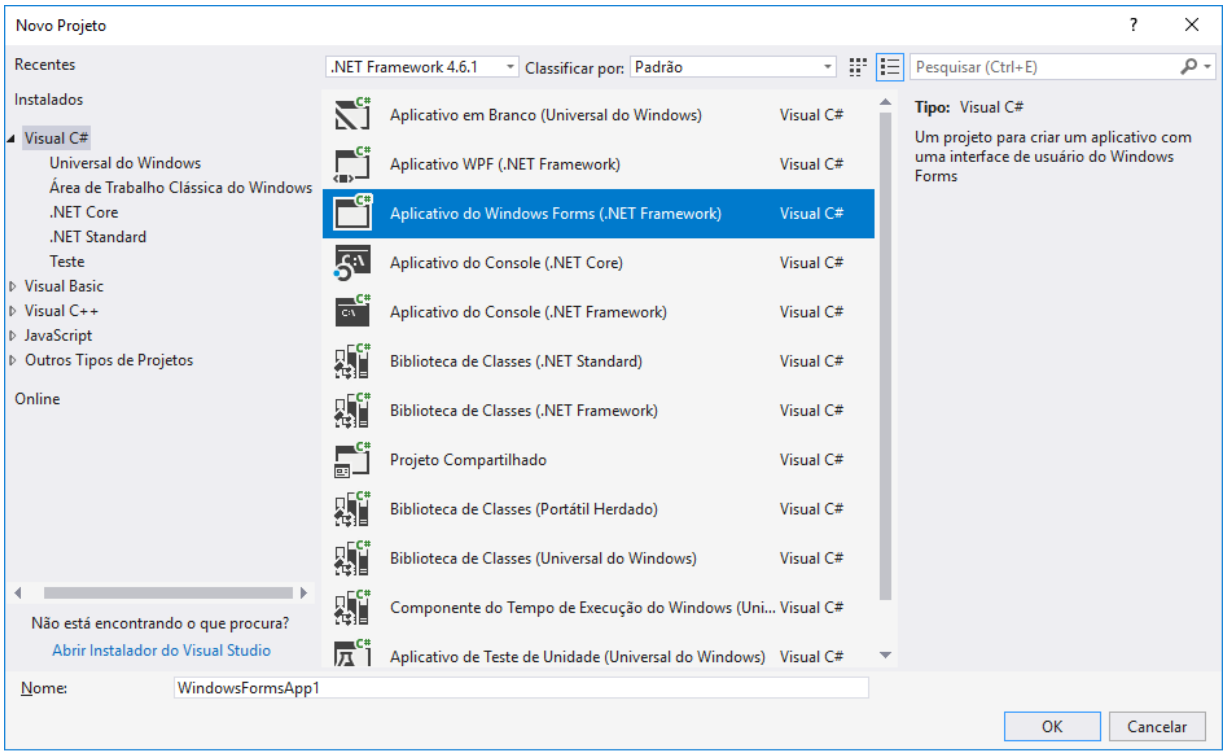
Fonte: Elaborado pela autora (2018).

2.5.4.1 Visual C#

Este trabalho utilizará o Visual C#, que pode ser considerado uma implementação propriamente dita da linguagem C#. No Visual C#, são oferecidas ao usuário ferramentas que auxiliam e aceleram o processo de desenvolvimento. Algumas dessas ferramentas são: um compilador, um editor de textos e um depurador avançado.

A Figura 17 mostra a tela de geração de um novo projeto, onde há todos os tipos de projetos que podem ser realizados no Visual Studio C# por intermédio da biblioteca .NET Framework 4.6.1.

Figura 17 – Tela para geração de um novo projeto C#



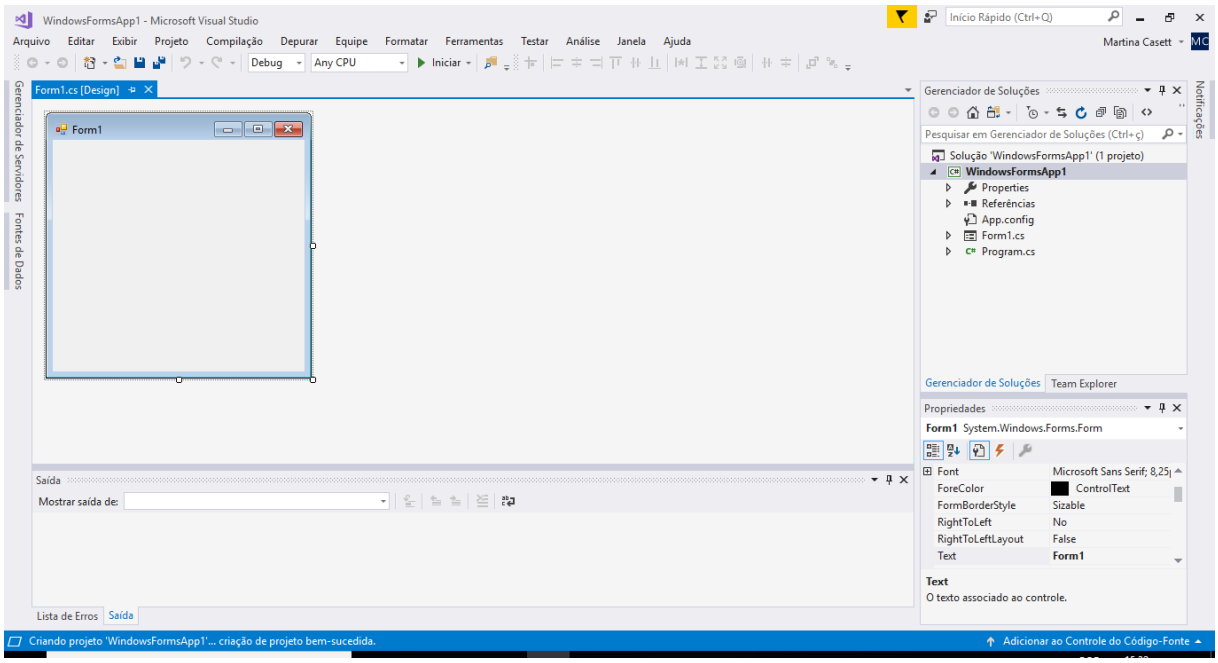
Fonte: Elaborado pela autora (2018).

2.5.4.2 Windows Form Application

A aplicação que será utilizada no Visual Studio é a *Windows Form Application,* compreendida no Visual C#. Essa aplicação é utilizada para implementar interfaces gráficas. Uma das vantagens inerentes à utilização dessa plataforma é a existência de inúmeras funções pré-programadas. Como exemplo dessas funções, pode-se citar a criação de um botão, a qual prescinde do desenvolvimento de um código, pois a programação já está previamente definida.

A Figura 18 fornece uma visão geral do que ocorre quando é selecionada a opção *Windows Form Application*.

Figura 18 – Página inicial do Windows Form Application



Fonte: Elaborado pela autora (2018).

3 DESENVOLVIMENTO

Nesta etapa será abordado o desenvolvimento do *software*, desde a criação do programa até a finalização do mesmo.

3.1 ATIVIDADES INICIAIS

Inicialmente, fez-se necessário realizar um estudo acerca da linguagem a ser utilizada, C#, do ambiente de desenvolvimento Visual Studio, e da sua plataforma Windows Form Application, na qual o programa foi desenvolvido e estruturado.

Para tal conhecimento, realizou-se a instalação do Microsoft Visual Studio no computador em que o projeto foi desenvolvido. Posteriormente, realizaram-se pesquisas em *sites* que implementam projetos similares, com intuito de obter ideias sobre como o tratamento dos dados de entrada do *software* seria efetuado.

3.2 CONVERSOR RS485

Após as atividades iniciais serem realizadas, compreendeu-se a importância do conhecimento a respeito do equipamento, o conversor STM-8522, utilizado para efetuar a comunicação dos dispositivos com o computador.

O processo de busca e envio de informações ocorre através da porta serial USB de um computador, onde a mensagem do mestre é enviada para essa porta. Assim a mensagem chega até os escravos, os quais identificam seu(s) destinatário(s). Após essa identificação, o dispositivo questionado irá responder, passando a resposta para todos os outros dispositivos, até chegar ao conversor, que irá converter de RS485 para USB. Em seguida, a mensagem é recebida pelo PC na porta serial, possibilitando a leitura da referida resposta pelo *software*.

Para que a rede funcione corretamente, alguns parâmetros do conversor RS485 devem ser configurados. De acordo com o manual de cada equipamento utilizado, as configurações da comunicação que devem ser formatas são: taxa de transmissão, data bits, paridade e bits de parada.

3.3 DESENVOLVIMENTO DO PROGRAMA

3.3.1 Biblioteca

Para que não ocorra algum tipo de erro de compilação, algumas bibliotecas já são inseridas pelo *software* quando o projeto é criado, e no decorrer da programação são adicionadas bibliotecas mais específicas, para que os elementos que necessitam destas possam ser utilizados sem ocorrências de falhas.

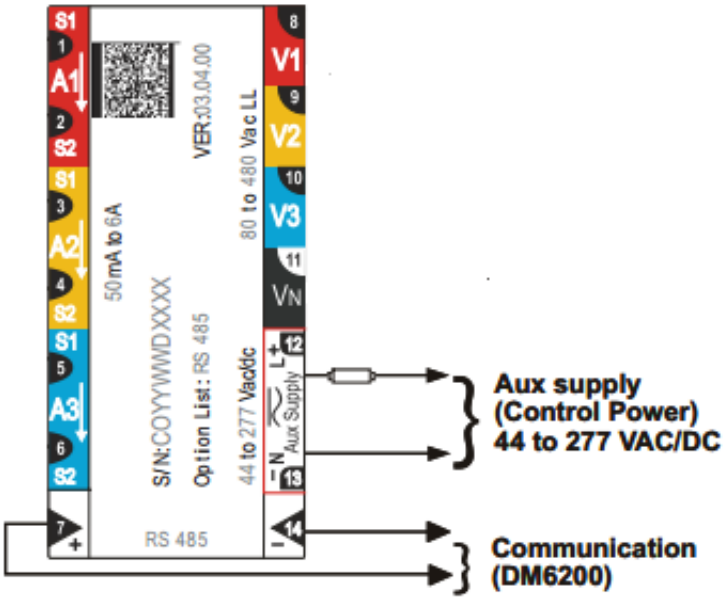
Neste projeto, as bibliotecas NModbus4 e NModbus.Serial, precisaram ser implementadas na programação, pois as mesmas fornecem conectividade a dispositivos e aplicativos compatíveis com o escravo MODBUS. Suportam protocolos ASCII, serial RTU, TCP e UDP.

Também foi implementada a biblioteca System.IO.Ports, a qual propõe-se a possibilitar a conexão com a porta USB.

3.3.2 Comunicação MODBUS com medidor de energia

Para realizar a comunicação serial do dispositivo com o *software*, primeiramente é necessária a conexão RS485 de dois fios na parte posterior no medidor de energia.

Figura 19 – Conexões MODBUS – RS485



Fonte: SCHNEIDER ELECTRIC (2010).

No *software* é criada uma função para atualizar a porta COM (Component Object Model), que será utilizada para o conversor RS485, com o propósito de viabilizar as devidas configurações de comunicação da porta serial, conforme Figura 20.

Figura 20 – Configuração da comunicação do medidor de energia

Digital meter Communication Settings	
Protocol	Modbus RTU
Data bits	8
Baud rate	9600 Baud, User set 1200 to 19200 Range:1200, 2400, 4800, 9600, 19200 Normal use: 9600 Baud Noisy, EMI, RFI, long data cable: 4800/2400 Baud Short cable (<300 meters or 975 feet): 19200 Baud
Parity	Even
Device Address	1
Stop bit	1
Modbus Protocol	
Device Address	1 to 247 Upto 247 meters per COM Port with Repeaters
Function Code	03 (Read)
Data Address	Refer to “Data address” on page 47 for more information.
Data type	32-bit float (real) : • All parameters. • Direct reading, little endian float, no scaling required 32-bit unsigned integer : • INTR (number of interruptions (outages) - RMS Blocks) • RunSec (Run seconds – Integ Block)
No of Registers	2 to 50 (optional) per DM6200 data block of 10 x 32 bit values must be configured to suit the digital meter

Fonte: SCHNEIDER ELECTRIC (2010).

Função para atualizar a porta a ser utilizada:

```
using (SerialPort port = new SerialPort("COM9"))
```

Configurações da comunicação:

```
port.BaudRate = 19200;
port.DataBits = 8;
port.Parity = Parity.Even;
port.StopBits = StopBits.One;
port.Open();
```

Para que o medidor possa funcionar de forma adequada e segura, são utilizados transformadores de corrente. Estes, além de prover isolação elétrica entre o primário (linha) e o secundário, possibilitam que o medidor opere dentro de suas especificações elétricas. Visto que os TCs operam dentro de sua classe de exatidão, a corrente no primário é proporcional[24] à corrente no secundário, sendo esta proporcionalidade definida como relação de transformação de corrente (RTC = I1/I2). Para os TCS utilizados a RTC é de 15, isto é, para um fluxo de corrente de 75 [A], induz-se no primário uma corrente de 5 [A]. Diante dessa transformação de corrente, faz-se imprescindível considerar a RTC na aquisição das grandezas elétricas pelo medidor de energia DM6200 da Schneider Electric.

3.3.2.1 Endereçamento

Na transição de uma mensagem MODBUS, há identificadores de início a fim de *framing*, com isso, é possível que os dispositivos da rede detectem o início de uma mensagem, leiam o seu campo de endereço e determinem qual dispositivo está sendo endereçado.

Nesta etapa é criada a chamada do mestre para identificação dos escravos. Além disso, são definidos os parâmetros de endereçamento dos dados a serem lidos. Esses endereços de parâmetros são encontrados no manual do medidor de energia, como demonstra o quadro 5 abaixo:

[24] Desconsiderando-se a parcela ínfima de correntes parasitas induzidas no núcleo, visto que se trata de operação em regime sem saturação magnética.

Quadro 5 – Endereço de parâmetro individual

Parameter	Description	Address	Type	DM6200
Metering				
Metering - Current				
A	Current Average	3913	Float	•
A1	Current, Phase 1	3929	Float	•
A2	Current, Phase 2	3943	Float	•
A3	Current, Phase 3	3957	Float	•
Metering – Voltage				
VLL	Line to Line Average Voltage	3909	Float	•
VLN	Line to neutral voltage	3911	Float	•
V12	Voltage phase1 to phase2	3925	Float	•
V23	Voltage phase2 to phase3	3939	Float	•
V31	Voltage phase3 to phase1	3953	Float	•
V1	Voltage phase1 to neutral	3927	Float	•
V2	Voltage phase2 to neutral	3941	Float	•
V3	Voltage phase3 to neutral	3955	Float	•
Metering – Power Factor				
PF	Power factor average	3907	Float	•
PF1	Power factor, phase1	3923	Float	•
PF2	Power factor, phase2	3937	Float	•
PF3	Power factor, phase3	3951	Float	•
Metering - Frequency				
F	Frequency, Hz	3915	Float	•
Integrator				
On hours	On hours	3993	Long	•
INTR	Number of power interruption	3999	Long	•
Percentage of Load parameters				
% Avg Load	Average Load percentage	3881	Float	•
%L1	Percentage of phase1 load	3883	Float	•
%L2	Percentage of phase2 load	3885	Float	•
%L3	Percentage of phase3 load	3887	Float	•
Unbalanced %Load	Unbalanced %Load	3889	Float	•
Unbalanced % Voltage	Unbalanced % Voltage	3891	Float	•

Fonte: SCHNEIDER ELECTRIC (2010).

O valor de endereço do escravo no protocolo MODBUS pode variar de 1 a 247, mas conforme a configuração do medidor de energia, o endereço utilizado para identificação é 1.

Função de chamada do mestre:

```
IModbusSerialMaster master = ModbusSerialMaster.CreateRtu(port);
```

Endereço de identificação do escravo a ser chamado:

```
byte slaveId = 1;
```

Endereços para a leitura de dados:

```
ushort v_a = 3924;
ushort v_b = 3938;
ushort v_c = 3952;
ushort frequencia = 3914;
ushort i_a = 3928;
ushort i_b = 3942;
ushort i_c = 3956;
ushort fp_a = 3922;
ushort fp_b = 3936;
ushort fp_c = 3950;
```

A palavra-chave *ushort*, indica um tipo de dados integrais que armazena valores de acordo com o tamanho (2 *bytes* ou 16 *bits*) e o intervalo de 0 a 65.535.

Através de alguns endereços, é possível calcular os valores das potências ativas, reativas e aparentes de cada fase. As fórmulas utilizadas são de um sistema trifásico com cargas não equilibradas.

Cálculos para obtenção do valor das potências:

```
var potenciaAparenteFase_a = v_a * i_a;
var potenciaAparenteFase_b = v_b * i_b;
var potenciaAparenteFase_c = v_c * i_c;

var potenciaAtivaFase_a = fp_a * potenciaAparenteFase_a;
var potenciaAtivaFase_b = fp_b * potenciaAparenteFase_b;
var potenciaAtivaFase_c = fp_c * potenciaAparenteFase_c;
```

A função Math.Acos é utilizada para poder realizar o cálculo do ângulo de cada fase a partir de seu respectivo fator de potência.

```
var anguloFase_a = Math.Acos(fp_a);
var anguloFase_b = Math.Acos(fp_b);
var anguloFase_c = Math.Acos(fp_c);
```

A potência reativa monofásica é obtida através da multiplicação da potência aparente desta fase pelo seno do seu respectivo ângulo.

```
var potenciaReativaFase_a = potenciaAparenteFase_a* Math.Sin(anguloFase_a);
```

```
var potenciaReativaFase_b = potenciaAparenteFase_a* Math.Sin(anguloFase_b);
var potenciaReativaFase_c = potenciaAparenteFase_a* Math.Sin(anguloFase_b);
```

Nesse contexto, a palavra-chave var, é uma variável local do tipo implícito, fortemente tipada, sendo determinada pelo próprio compilador.

As variáveis que são lidas no medidor de energia são do tipo *float* - 4 *bytes*, portanto, o programa interpretará essas variáveis como sendo do tipo *float*.

3.3.2.2 Código de Função MODBUS

Na especificação do protocolo MODBUS RTU, são definidas as funções utilizadas para acessar os tipos de registradores descritos na especificação. Para o medidor de energia, tanto os parâmetros quanto as variáveis básicas foram definidos como registradores do tipo *holding*.

3.3.2.2.1 Função 03 – Read Holding Register

Esta função lê o conteúdo de um grupo de registradores, que necessariamente devem estar em sequência numérica. No quadro 6, é possível visualizar a estrutura para os telegramas de leitura e resposta (cada campo representa um *byte*).

Quadro 6 – Função Read Holding Register

Pergunta (Mestre)	**Resposta (Escravo)**
Endereço do escravo	Endereço do escravo
Função	Função
Endereço do registrador inicial (byte high)	Campo Byte Count
Endereço do registrador inicial (byte low)	Dado 1 (high)
Quantidade de registradores (byte high)	Dado 1 (low)
Quantidade de registradores (byte low)	Dado 2 (high)
CRC-	Dado 2 (low)
CRC+	etc...
	CRC-
	CRC+

Fonte: WEG (2011).

Mensagem do mestre para o escravo

```
registersReaded = master.ReadHoldingRegisters(slaveId, v_a, 2);
v_a = Modbus.Utility.ModbusUtility.GetSingle(registersReaded[1],
registersReaded[0]);
```

Nesta mensagem o mestre solicita o ID do escravo, os endereços e o número de registradores a serem lidos. A quantidade de registros é definida nas funções da biblioteca MODBUS.

Os dados lidos são representados por variáveis tipo *float*, formadas por 32 *bits*, tornando necessária a leitura de dois registros para armazenar um único valor de dados. Cada um dos registradores registersReaded[1] e registersReaded[0] é responsável pelo armazenamento de 2 *bytes*, os quais irão receber a informação da variável *float* a ser lida, onde o primeiro *byte* enviado é o menos significativo (LSB) e é seguido pelo *byte* mais significativo (MSB).

3.3.2.3 Resultado

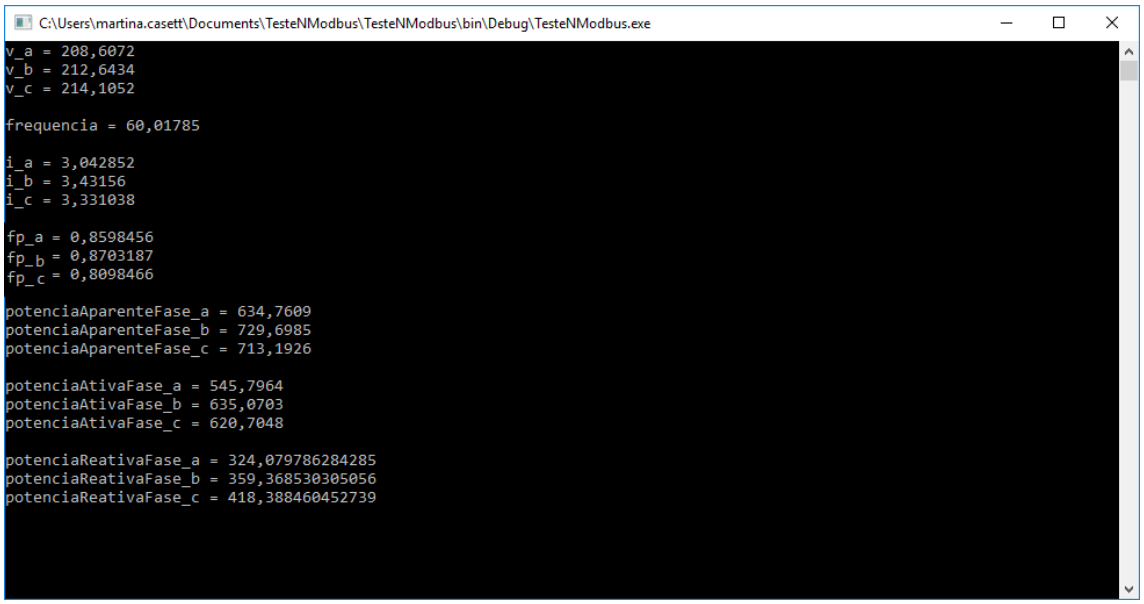
Mediante a correta configuração dos itens, tem-se a análise de alguns dados coletados pelo analisador de energia.

Neste teste um motor de indução trifásico foi conectado ao analisador de energia, proporcionando mensurar, direta ou indiretamente, as seguintes grandezas elétricas: tensões (entre fase e neutro), correntes, frequência elétrica, fatores de potência[25], e as respectivas potências aparentes, ativas e reativas de cada fase.

Os dados obtidos pelo *software*, apresentados na Figura 21, referem-se à operação sob carga nominal de um motor de indução trifásico de 2 [CV]. Os valores mensurados estão de acordo com o que se esperava. Percebe-se que, apesar de os fatores de potência de cada uma das 3 fases apresentarem valores distintos (carga levemente desbalanceada), todos ficaram muito próximos do fator de potência nominal, isto é, aproximadamente 0,85 indutivo (≈ 60º). Os dados obtidos demonstram êxito no endereçamento e configuração do portal serial e, portanto, a viabilidade da comunicação de dados entre o analisador de energia e o computador por meio do protocolo MODBUS RTU.

[25] Dado que a carga é desbalanceada.

Figura 21 – Resultado de dados coletados do Analisador Energia

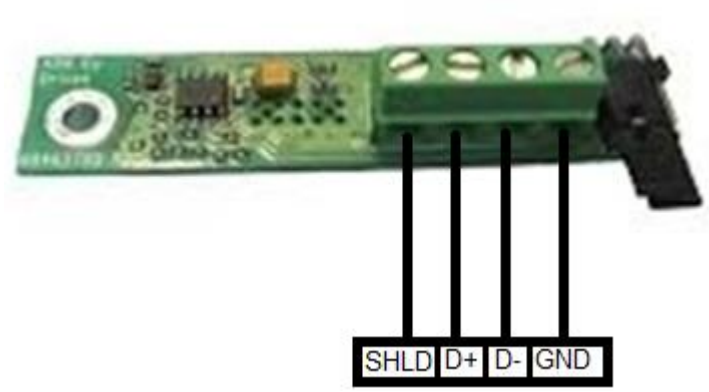


Fonte: Elaborado pela autora (2018).

3.3.3 Comunicação MODBUS com Inversor de Frequência

Para a comunicação serial, o inversor ACS355, necessita de um módulo de comunicação serial. Neste trabalho, será utilizado o FRSA-00 que, ao ser inserido na parte frontal do inversor de frequência, estabelecerá um caminho para o fluxo de dados por meio da conexão RS485.

Figura 22 – Conexão MODBUS – RS485



Fonte: Elaborado pela autora (2018).

De forma distinta do que ocorre com medidor de energia, o inversor de frequência deve ser parametrizado manualmente através de sua IHM. Inicialmente, a

parametrização para habilitar o controle via interface serial é realizada. Em seguida, parametrizam-se o ID do escravo e os endereços para coleta de dados.

Visto que o propósito do presente trabalho é comunicar dispositivos em uma mesma rede, recomenda-se que os parâmetros de configuração para porta serial sejam os mesmos para todos, evitando assim possíveis erros na comunicação.

O primeiro parâmetro necessário para que se possa iniciar a comunicação MODBUS é o parâmetro 9802, sendo definido o valor 1.

Quadro 7 – Parâmetro MODBUS

98 OPÇÕES	Ativação da comunicação série externa	
9802 SEL PROT COM	Ativa a comunicação série externa e seleciona o interface.	*NÃO SEL*
NÃO SEL	Sem comunicação	0
MODBUS STD	Fieldbus integrado. Interface: EIA-485 fornecido pelo Adaptador Modbus FMBA-01 opcional ligado ao terminal X3 do conversor. Veja o capítulo *Controlo por fieldbus com fieldbus integrado* na página *337*.	1

Fonte: SCHNEIDER ELECTRIC (2010).

Em seguida, os parâmetros 5302, 5303, 5304 e 5305 são alterados de acordo com as configurações contidas no quadro 8. O endereço de identificação do inversor foi definido como 3, a taxa de transmissão e paridade foram definidos uniformemente com o medidor de energia.

Quadro 8 – Configuração da comunicação do Inversor de Frequência

CONFIGURAÇÃO DO MÓDULO ADAPTADOR			
5302 ID ESTAÇÃO EFB	0…247	Qualquer	Define a velocidade de comunicação da ligação RS-232/EIA-485. Não é possível duas estações em linha com o mesmo endereço.
5303 TAXA TRANSM EFB	1.2 kbit/s 2.4 kbit/s 4.8 kbit/s 9.6 kbit/s 19.2 kbit/s 38.4 kbit/s 57.6 kbit/s 115.2 kbit/s		Define a velocidade de comunicação da ligação RS-232/EIA-.
5304 PARIDADE EFB	*8 NONE 1* *8 NONE 2* *8 PAR 1* *8 IMPAR 1*		Seleciona o ajuste da paridade. Devem usar-se as mesmas definições em todas as estações em linha.
5305 CTRL PERFIL EFB	*ABB DRV LIM* *PERFIL DCU* *ABB DRV CPL*	Qualquer	Seleciona o perfil de comunicação usado pelo conversor. Veja a secção *Perfis de comunicação* na página *353*.
5310 PAR 10 EFB … *5317 PAR 17 EFB*	0…65535	Qualquer	Seleciona um valor atual para ser mapeado para o registo Modbus 400xx.

Fonte: SCHNEIDER ELECTRIC (2010).

3.3.3.1 Endereçamento

Para que o inversor colete os dados de forma apropriada, alguns parâmetros devem ser configurados conforme o que é apresentado no manual do equipamento.

Os parâmetros que podem ser mapeados para o registro MODBUS vão do 5310 ao 5317.

Quadro 9 – Parâmetros setados

Parâmetro	Registro Modbus	Valor setado	Descrição
5310	40005	103	Frequência Saída [Hz]
5311	40006	104	Corrente [A]
5312	40007	102	Velocidade Motor [rpm]
5313	40008	107	Tensão CC [V]
5314	40009	109	Tensão CA [V]
5315	40010	110	Temperatura IGBT [°C]
5316	40011	401	Ultima falha
5317	40012	412	Penúltima falha

Fonte: Elaborado pela autora (2018).

Os parâmetros de referências e valores atuais do inversor de frequência são mapeados com área 4xxxx. Na mensagem de dados MODBUS, esse registro é endereçado como xxxx-1. Os registros MODBUS que são utilizados para selecionar um valor atual a ser mapeado vão do 40005 ao 40012.

```
ushort frequencia = 0004;
ushort correnteMotor = 0005;
ushort velocidadeMotor = 0006;
ushort tensaoCC = 0007;
ushort tensaoCA = 0008;
ushort tempIGBT = 0009;
ushort ultimaFalha = 0010;
ushort penultimaFalha = 0011;
```

3.3.3.2 Código de Função MODBUS

A função utilizada para os inversores também foi a função 03 – *read holding register*.

Mensagem do mestre para o escravo, solicitando ID do escravo, endereços e registros a serem lidos:

```
var registersReaded = master.ReadHoldingRegisters(slaveId, frequencia, 1);

var frequencia = (float)registersReaded[0];
```

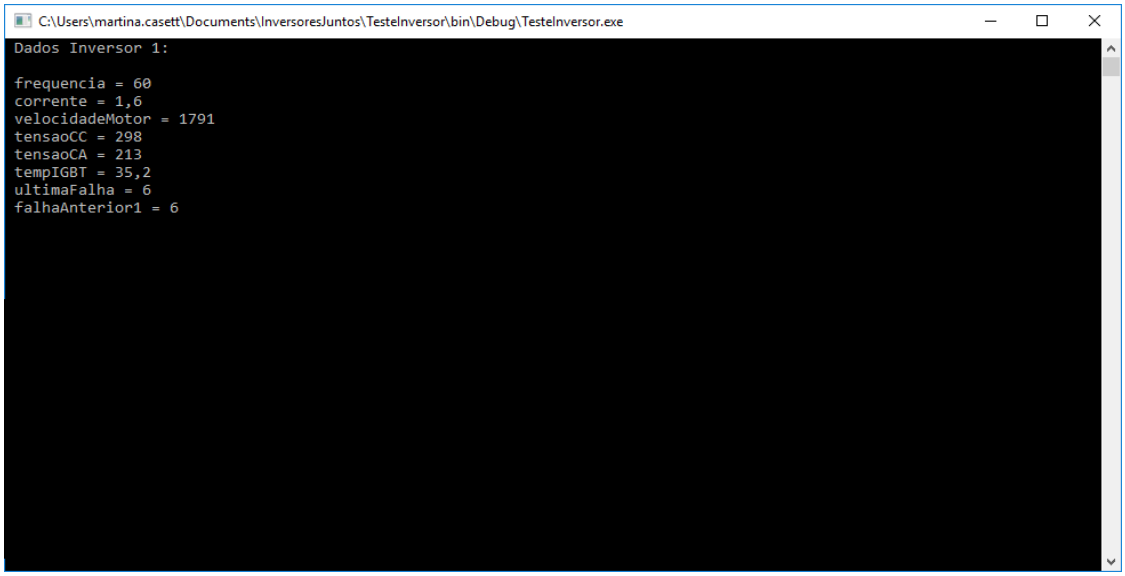
Os parâmetros da função de leitura de dados da biblioteca MODBUS utilizada necessitam de parâmetros diferentes para a leitura de *floats* em cada equipamento, isso ocorre devido às características do mesmo.

A comunicação entre um sistema MODBUS e um inversor de frequência consiste em palavras de dados de entrada e de saída de 16 bits (2 *bytes*), requerendo o uso de apenas um registrador.

3.3.3.3 Resultado

O inversor de frequência, assim como o medidor de energia elétrica, é passível de conexão pela rede serial RS485, podendo coletar inúmeros dados, identificar falhas e acionar alarmes, como mostra a figura 23.

Figura 23 – Dados coletados do Inversor de Frequência

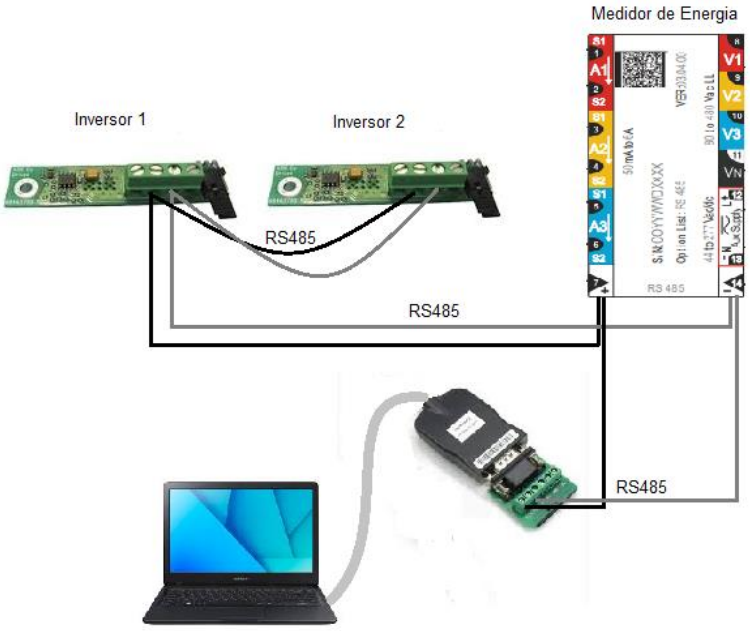


Fonte: Elaborado pela autora (2018).

3.3.4 Comunicação global MODBUS

A conexão física entre os dispositivos é realizada através do módulo MODBUS, que consiste no inversor, munido de fios para comunicação MODBUS do medidor de energia, e conversor RS485, conforme figura 24.

Figura 24 – Conexão entre dispositivos



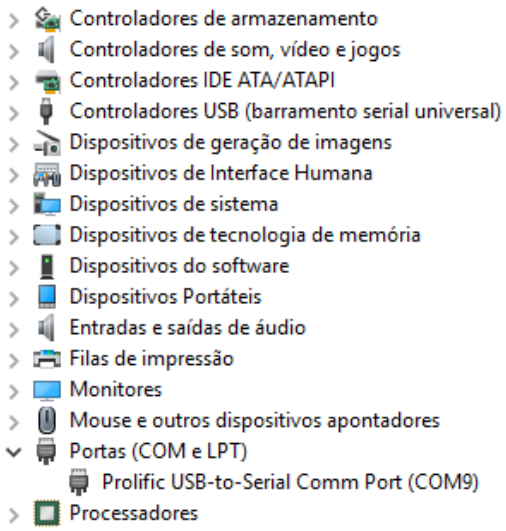
Fonte: Elaborado pela autora (2018).

O conversor RS485, assim como os inversores de frequência e o medidor de energia, foi configurado para atuar na mesma rede MODBUS. Para isso, deve-se acessar o gerenciador de dispositivos do Windows e configurar a taxa de comunicação para 19200 [bps], 8 bits de dados, paridade par (*even*) e 1 *stopbit*.

3.3.4.1 Configuração do Conversor RS485 - USB

O primeiro passo para configurar o conversor RS485-USB é acessar o “Gerenciador de dispositivos” do Windows e clicar duas vezes em “Prolific USB-to-Serial Comm Port”, como ilustra a Figura 25.

Figura 25 – Localização do cabo Conversor

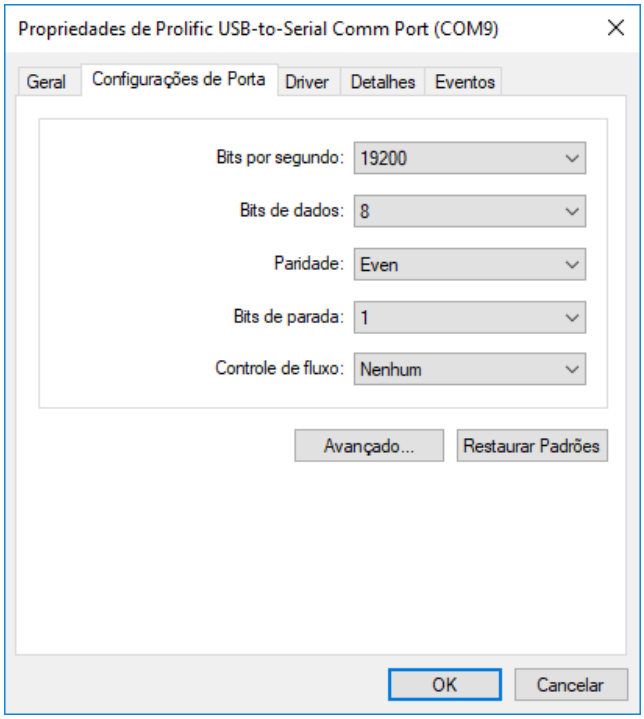


Fonte: Elaborado pela autora (2018).

Após clicar em “Prolific USB-to-Serial Comm Port (COM9)”, aparecerá a janela da figura 26.

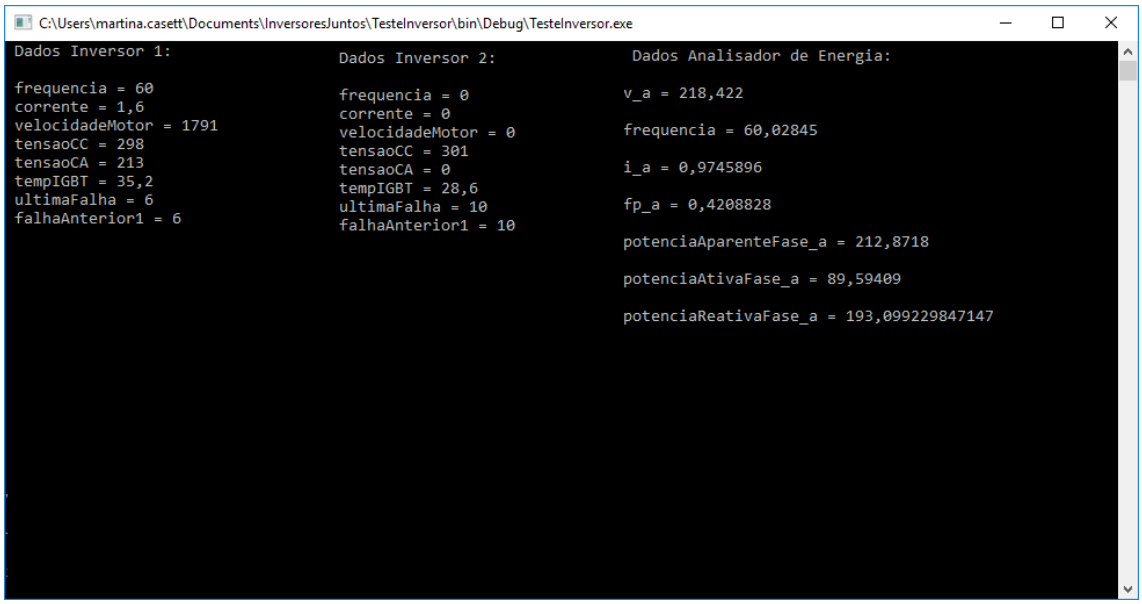
Na aba “Configurações de porta” devem ser modificados os campos bits por segundo [*baud rate*], bits de dados, método de paridade e bits de parada, em conformidade com o que consta no manual de cada dos equipamentos utilizados.

Figura 26 – Propriedades de Prolific USB-to-Serial Comm Port



Fonte: Elaboração da autora (2018)

Figura 27 – Resultado dos dados coletados dos três equipamentos



Fonte: Elaborado pela autora (2018).

Nestes testes foram utilizados um inversor de frequência (“Inversor 1” na Figura 27) acoplado a um motor de indução trifásico operando em vazio (sem carga); um analisador de energia e um outro inversor de frequência (“Inversor 2”). A tensão na entrada do “Inversor 1” é de 220 [V] (fase-neutro), já que o teste foi realizado utilizando uma tomada monofásica convencional. Podemos observar que:

$$V_{open_{FN}} = V_{nom_{FN},motor} \approx 220\ [V]$$

$$s = \frac{ns-n}{ns}.100\% = \frac{1800-1791}{1800}.100\% = 0{,}5\ \% \qquad (12)$$

$$Q_{open} > P_{open}$$

Os dados apresentados na figura 27 corroboram com o que ensina a teoria acerca dos motores trifásicos de indução operando em vazio: mantendo-se o rotor livre (sem carga mecânica) ao passo em que se aplica tensão nominal ao enrolamento primário (estator), observa-se em um escorregamento próximo de zero (s ≈ 0,5%), isto é, a velocidade de rotação no eixo do motor [rpm] é aproximadamente igual à velocidade de rotação do campo magnético girante no entreferro. Ainda, percebe-se que o fator de potência (fp_a = 0,42), é baixo e

indutivo, o que era previsto, uma vez que a energia fornecida pela rede elétrica não realiza trabalho útil.

A redução significativa do fator de potência na operação em vazio é amplamente abordada na literatura. Em relação a este ensaio, a baixa potência ativa consumida pelo motor (e medida pelo medidor) pode ser explicada, primordialmente, por: dissipação na forma de calor (Efeito Joule) nas bobinas de cobre do estator; perdas por histerese e circulação de correntes parasitas induzidas nas chapas do motor; perdas devido ao atrito e à ventilação associados à rotação do motor. Ao contrário do que ocorre com a parcela de potência ativa, na operação em vazio, a potência reativa indutiva fornecida não diminui muito e, via de regra, Qvazio > Pvazio.

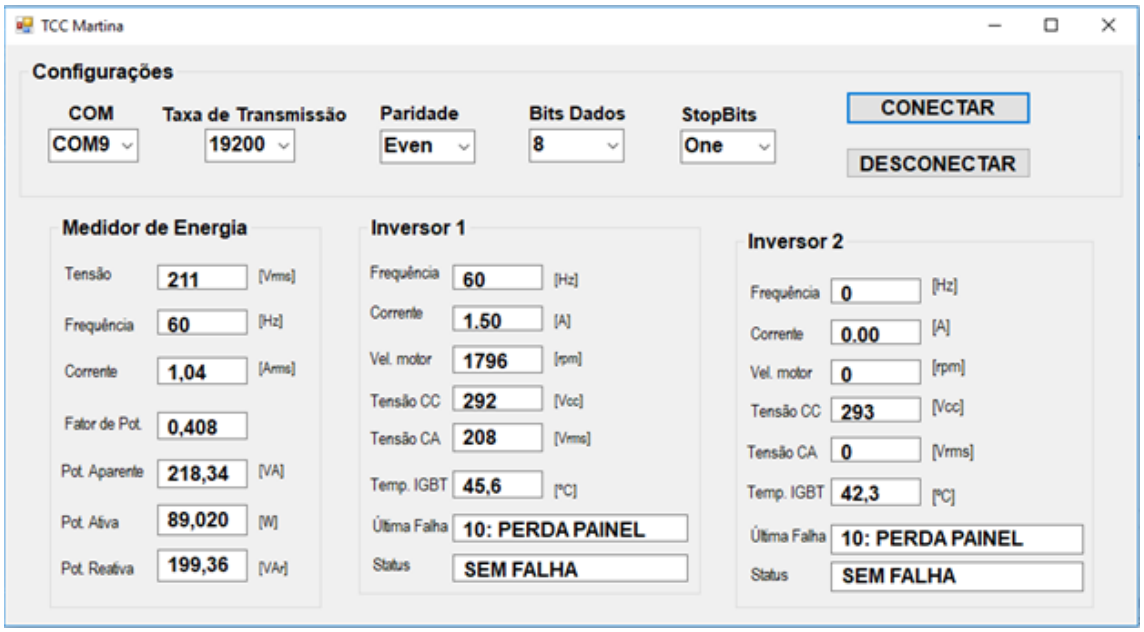
Por fim, nota-se que a troca de informações entre o “Inversor 1” e o “Inversor 2” também ocorreu de forma satisfatória. Vale ressaltar que esse último, isolado da carga elétrica (motor), foi inserido no sistema apenas para comprovar a viabilidade de que dados fossem trocados entre dois ou mais inversores do mesmo sistema pelo protocolo MODBUS RTU por intermédio do padrão físico RS485.

3.3.5 Interface gráfica

Uma boa interface complementa adequadamente o potencial de um programa computacional, tornando a interação com o mesmo agradável e intuitiva. Dessa forma, otimiza-se a experiência do usuário, além de facilitar a tomada de decisões em resposta à visualização dos resultados obtidos.

Para construir a interface gráfica, é preciso escolher os componentes desejados na caixa de ferramentas (*toolbox*), organizando-os da forma desejada.

Figura 28 – Interface gráfica para coleta de dados



Fonte: Elaborado pela autora (2018).

Foram utilizadas as seguintes ferramentas para a construção da interface:

groupBox: para otimizar a organização e tornar a visualização mais amigável ao usuário;
labels: denominam os dados que estão sendo representados;
textBox: informam os valores dos dados adquiridos pelos equipamentos;
comboBox: permitem selecionar a configuração da porta serial mais adequada para uma determinada operação;
button: acionam e interrompem a comunicação entre o mestre e o(s) escravo(s).

A coleta de dados é realizada em tempo real (uma vez a cada segundo), para que isso se torne possível, é necessário utilizar a ferramenta *timer* do Windows Forms Application.

4 CONCLUSÃO

Este trabalho foi baseado na integração de tecnologias distintas presentes em redes de automação, através da compatibilização das redes tanto em nível de *hardware*, nomeadamente em termos de interfaces, conversores e meios físicos, quanto em nível de *software*, através da compatibilização dos protocolos de comunicação.

O presente projeto procurou validar a comunicação entre dispositivos da máquina de corte, utilizando a arquitetura mestre-escravo, através de uma rede industrial, na qual os bits são enquadrados e codificados conforme o protocolo MODBUS-RTU, utilizando o padrão serial RS-485.

Para auxiliar o *software* de corte da máquina, reduzindo sua demanda de processamento de dados e da IHM, fez-se necessária a criação de outro *software,* cujo desenvolvimento ocorreu no ambiente Visual Studio, utilizando a linguagem de programação C# e o Windows Forms Application para interface gráfica.

Os objetivos fundamentais foram alcançados com sucesso: o desenvolvimento do *software* foi realizado com êxito e a comunicação MODBUS entre os equipamentos procedeu de acordo com o que se esperava, fornecendo, através da interface gráfica, dados fidedignos àqueles obtidos por intermédio de outros instrumentos de medição. A análise dos dados coletados por intermédio do *software* proporcionará, entre outras coisas, a realização de manutenções preditivas na máquina. Por conseguinte, diminuindo a ocorrência de falhas nos equipamentos que a constituem e possibilitando resultados satisfatórios para os clientes.

REFERÊNCIAS

ALBUQUERQUE, Pedro Urbano Braga de; ALEXANDRIA, Auzuir Ripardo de. **Redes Industriais:** Aplicações em Sistemas Digitais de Controle Distribuído. Curitiba: Editora Livro Técnico, 2009.

ANEEL. **Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional – PRODIST**. 2017. Disponível em: <http://www.aneel.gov.br/documents/656827/14866914/Módulo_8-Revisão_10/2f7cb862-e9d7-3295-729a-b619ac6baab9>. Acesso em: 22 mar. 2018.

ASTRÖM, Karl J.; WITTENMARK, Björn. **Computer–Controlled Systems**: Theory and Design. 3. ed. Prentice–Hall, 1997.

AZEVEDO, Julio Arlindo Pinto; SOUZA, Alex Bernardes de. **Comparativo entre redes de automação industrial e suas características**. 2017. Disponível em: <http://www.inatel.br/biblioteca/pos-seminarios/seminario-de-automacao-industrial-e-sistemas-eletro-eletronicos/i-saisee/9383-comparativo-entre-redes-de-automacao-industrial-e-suas-caracteristicas/file>. Acesso em: 19 mar. 2018.

BECK, Kent. **Extreme Programming Explained**: Embrace Change. 2. ed. Massachusetts: Addison-Wesley, 2004.

BEGA, Egídio Alberto (Org.). **Instrumentação Industrial.** 3. ed. Rio de Janeiro: Ed. Interciência, 2011.

BIRD, John. **Circuitos Elétricos**. 3. ed. São Paulo: Elsevier Editora, 2009.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML:** Guia do Usuário. 2. ed. São Paulo: Editora Campus, 2005.

CARCANO, Andrea. **Advanced security aspects on Industrial Control Network**. 2013. 139 f. Tese (Phd em Ciências da Computação) – Departamento de Ciência Teórica e Aplicada, Università degli Studi dell'Insubria, Varese, Itália. Disponível em: <http://insubriaspace.cineca.it/bitstream/10277/509/1/Phd_thesis_carcanoandrea_completa.pdf>. Acesso em: 20 mar. 2018.

CHAPMAN, Stephen J. **Electric Machinery Fundamentals**. 5. ed. McGraw-Hill, 2013.

CREDER, Hélio. **Instalações Elétricas**. 16. ed. Rio de Janeiro: LTC, 2016.

FOX, Robert W.; PRITCHARD, Philip J.; MCDONALD, Alan T. **Introdução à Mecânica dos Fluidos**. 8. ed. São Paulo: LTC, 2011.

FRENZEL JUNIOR, Louis E. **Handbook of serial communications interfaces:** a comprehensive compendium of serial digital input/output (I/O) standards. 1. ed. Walthan: Elsevier, 2016.

FRANCHI, Claiton Moro. **Acionamentos elétricos**. 4. ed. São Paulo: Editora Érica, 2010.

FUKUYOSHI, Bruna Hitomi. **Programa para Gerenciamento de Rebanho Leiteiro**. 2012. 55 p. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2012.

GALLOWAY, Brendan; HANCKE, Gerhard P. Introduction to Industrial Control Networks. **IEEE Communications Surveys & Tutorials**, v. 15, n. 2, jul. 2012**.**

HAYKIN, Simon. **Sistema de Comunicação:** analógicos e digitais. 4. ed. Porto Alegre: Editora Bookman, 2004.

IBM. **Rational Unified Process**: Best Practices for Software Development Teams. Disponível em: <https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>. Acesso em: 01 maio 2018.

KARNAM, Naresh. **MODBUS Implementation Using a Serial Link**. 2007. 74 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade do Tennessee-Knoxville. Disponível em: <http://trace.tennessee.edu/utk_gradthes/154>. Acesso em: 20 mar. 2018.

KINDERMANN, Geraldo. **Proteção de Sistemas Elétricos de Potência**. 3. ed. Florianópolis: UFSC/EEL/LABPLAN, 2012.

LOPEZ, Ricardo Aldabó. **Sistemas de Redes para Controle e Automação**. São Paulo: Editora Book Express, 2000.

LUGLI, Alexandre Baratella; SANTOS, Max Mauro Dias. **Redes Industriais para Automação Industrial:** AS-I, PROFIBUS e PROFINET**.** 1. ed. São José dos Campos: Editora Érica, 2010.

_____. Redes Industriais: Evolução, Motivação e Funcionamento. **Revista Intech América do Sul**, 2011. Disponível em: <http://www.inatel.br/biblioteca/artigos-cientificos/2011/8794-redes-industriais-evolucao-motivacao-e-funcionamento/file>. Acesso em: 19 mar. 2018.

MAMEDE FILHO, João. **Instalações Elétricas Industriais**. 8. ed. Rio de Janeiro: LTC, 2015.

MANIFESTO Ágil. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 01 maio 2018.

MICROSOFT. **Guia de C#**. 2018. Disponível em: <https://msdn.microsoft.com/pt-br/library/kx37x362.aspx>. Acesso em: 25 mar. 2018.

MICROSOFT. **Introdução ao Desenvolvimento com o Visual Studio**. 2015. Disponível em: <https://msdn.microsoft.com/pt-br/library/ms165079.aspx>. Acesso em: 24 mar. 2018.

MODBUS. **MODBUS Application Protocol Specification.** 2006b. v. 1.1b, dez. 2006. Disponível em: <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf>. Acesso em: 19 mar. 2018.

OSORIO, Jorge A.; CHAUDRON, Michel R. V.; HEIJSTEK, Werner. Moving From Waterfall to Iterative Development – An Empirical Evaluation of Advantages, Disadvantages and Risks of RUP. In: CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS – EUROMICRO, 37ª, 2011, Oulu. **Anais…** Oulu: IEEE Computer Society, 2011, p.453-460.

PETRUZELLA, Frank D. **Eletrônica II**. 1. ed. McGraw-Hill, 2014.

POWELL, James. **Profibus and Modbus**: A comparison. 2013. Disponível em: <https://www.industry.siemens.com/datapool/industry/automation/Tech-Art/2013/FAV-105-2013-IA-SC/FAV-105-2013-IA-SC-V01_EN.pdf>. Acesso em: 20 mar. 2018.

PRESSMAN, Roger S. **Engenharia de Software**. 6. Ed. McGraw-Hill, 2006.

_____. MAXIM, Bruce R. **Engenharia de Software**. 8. ed. McGraw-Hill, 2012.

RIZZONI, Giorgio. **Principles and applications of Electrical Engineering**. 1. ed. Mc Graw Hill, 2009.

SANCHES, Durval. **Especificação de Componentes**. Disponível em: <www.agbook.com.br>. Acesso em: 31 maio 2018.

SCHNEIDER ELECTRIC. **Easy Logic DM6000 Series Digital Meter User Guide**. 2015. Disponível em: <http://download.schneider-electric.com/files?p_Reference=NHA1696405&p_EnDocType=User guide&p_File_Id=1156694720&p_File_Name=NHA1696405-02.pdf>. Acesso em: 23 mar. 2018.

SCHNEIDER ELECTRIC. **Power Logic DM6000 Series Digital Meters:** User Manual. 2010. Disponível em: <http://www2.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/212000/FA212509/en_US/DM6200_User_Guide.pdf>. Acesso em: 09 maio 2018.

SCHNEIDER ELECTRIC. **Redes de comunicação industrial**: Documento técnico nº 2. 2007. Disponível em: <https://www.schneider-electric.pt/documents/product-services/.../doctecnico_redes.pdf>. Acesso em: 21 mar. 2018.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum – Um guia definitivo para o Scrum:** As regras do jogo. 2001. Disponível em: <https://contas.tcu.gov.br/etcu/ObterDocumentoSisdoc?seAbrirDocNoBrowser=true&codArqCatalogado=6271614>. Acesso em: 15 jun. 2018.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Pearson Prentice-Hall, 2000.

_____. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice-Hall, 2003.

_____. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice-Hall, 2011.

TAMAKI, Paulo Augusto Oyamada. **Uma extensão do RUP com ênfase no gerenciamento de projetos do PMBoK baseada em Process Patterns**. São Paulo. 2007. 211 f. Dissertação (Mestre em Engenharia) – Escola Politécnica da Universidade de São Paulo, Universidade de São Paulo, 2007. Disponível em: <file:///C:/Users/ASILVA27/Downloads/Dissertacao.pdf>. Acesso em: 14 jun. 2018.

UMANS, Stephen D. **Máquinas Elétricas de Fitzgerald e Kingsley**. 6. ed. Nova Iorque: McGraw Hill, 2006.

WEG. **MODBUS RTU** – Manual do Usuário. 2011. Disponível em: <http://ecatalog.weg.net/files/wegnet/WEG-plc300-comunicacao-modbus-rtu-10000850708-manual-portugues-br.pdf>. Acesso em: 09 maio 2018.

WILLE, Christopher. **Apresentando C#:** Um novo conceito de linguagem de programação orientada a objeto. São Paulo: Berkeley, 2001.