# DESENVOLVIMENTO DE SOFTWARE PARA PARAMETRIZAÇÃO, MONITORAMENTO E SUPERVISÃO DE CONVERSORES DE FREQUÊNCIA

**MARCELO CHRISTIANO DA FRANÇA JÚNIOR**

**CAMPO GRANDE**

**2011**

**UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL**

**PROGRAMA DE PÓS-GRADUAÇÃO**

**EM ENGENHARIA ELÉTRICA**

# Desenvolvimento de software para parametrização, monitoramento e supervisão de conversores de frequência

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica – Área de Concentração: Energia – da Universidade Federal de Mato Grosso do Sul, como parte para obtenção do título de mestre em Engenharia Elétrica, elaborada sob orientação do Profº Dr. Valmir Machado Pereira.

**MARCELO CHRISTIANO DA FRANÇA JÚNIOR**

**CAMPO GRANDE**

**2011**

# DESENVOLVIMENTO DE SOFTWARE PARA PARAMETRIZAÇÃO, MONITORAMENTO E SUPERVISÃO DE CONVERSORES DE FREQUÊNCIA


## MARCELO CHRISTIANO DA FRANÇA JÚNIOR


‘Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em *Energia*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campo Grande.’

_____

Valmir Machado Pereira, Dr.

Orientador

_____

Luciana Cambraia Leite, Dra.

Coordenadora do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Valmir Machado Pereira, Dr.

Presidente

_____

Edson Antonio Batista, Dr.

_____

Luciano Gonda, Dr.

*"Matar o sonho é matarmo-nos. É mutilar a nossa alma. O sonho é o que temos de realmente nosso, de impenetravelmente e inexpugnavelmente nosso"*

*(Fernando Pessoa)*

# AGRADECIMENTOS

Agradeço aos meus pais Marcelo e Marisa por tudo que representam em minha vida, me ajudando em minhas escolhas, sempre apoiando em minhas decisões.

A minha esposa Cristiane, pois graças a ela consegui concluir esta fase de minha vida graças ao seu apoio e carinho nas horas mais difíceis.

Ao meu orientador Valmir por toda sua paciência durante minha orientação e pelos seus sábios conselhos durante esta jornada.

Ao professor Edson que sempre deu idéias muito boas e se fez presente sempre que necessário.

Ao meu colega Guilherme, mais que um colega, amigo de muitos anos que me ajudou quando foi preciso e sempre disse que conseguiríamos alcançar nossos objetivos.

Ao meu colega Leandro por todo seu apoio durante este mestrado me incentivando nas horas mais difíceis.

A CAPES pelo apoio financeiro durante minhas pesquisas.

Ao DEL/UFMS pelo material cedido para os testes de validação do software.

Resumo da Dissertação apresentada a UFMS como parte dos requisitos necessários
para a obtenção do grau de Mestre em Engenharia Elétrica.

# Desenvolvimento de software para parametrização, monitoramento e supervisão de conversores de frequência


## MARCELO CHRISTIANO DA FRANÇA JÚNIOR



Maio / 2011

Orientador: Dr. Valmir Machado Pereira

Área de Concentração: Energia.

Palavras-chave: softwares supervisórios, conversor de freqüência, programação.

Número de Páginas: 79



RESUMO: A utilização de software supervisório é fundamental na automação de processos de diversas áreas, como indústria automotiva, robótica, agrícola, entre outros. Estes softwares supervisórios devem refletir as operações de diferentes tipos de sensores e atuadores. A maioria dos softwares supervisórios são proprietários, ou seja, possuem protocolo de comunicação específico para os transdutores do mesmo fabricante. Com isto o presente trabalho desenvolveu um software para monitoramento e atuação em um motor de indução trifásico utilizando um inversor para se obter a potência necessária para estas atividades entre o motor e o sistema de monitoramento. Este software foi desenvolvido com base em pesquisa relacionando as necessidades do projeto com a facilidade de manuseio de algumas linguagens de programação, relação feita também com o sistema operacional a ser utilizado. Os resultados apresentados neste trabalho, mostram a possibilidade de agregar funções em um sistema de automação, além de possuir recursos para fins didáticos.

Abstract of Dissertation presented to UFMS as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

# Development of software for parameterization, monitoring and supervision of frequency converters

**MARCELO CHRISTIANO DA FRANÇA JÚNIOR**

May / 2011

Advisor: Dr. Valmir Pereira Machado
Concentration Area: Energy.
Keywords: supervisory software, frequency inverter, programming.
Number of Pages: 79

ABSTRACT: The use of monitoring software is crucial in the automation of processes in various areas such as automotive, robotics, agriculture, among others. These supervisory software should reflect the operations of different types of sensors and actuators. Most software Supervisory owners are, or have communication protocol specific to the transducers from the same manufacturer. With this the present work has developed a software for monitoring and performance in an engine using a three-phase inverter to obtain the necessary power for these activities between the engine and system monitoring. This software was developed based on research needs relating project with the ease of handling of some programming languages, relationship also made with the operating system being used. The results presented in this paper show the possibility of adding functions in a automation system and also has resources for teaching purposes.

# Lista de Figuras

Figura 2.1: Módulo de comunicação RS-232 ..... 22
Figura 2.2: Exemplo de telegrama de leitura. ..... 24
Figura 2.3: Exemplo de telegrama de escrita. ..... 25
Figura 2.4: Combinação de campos para criação de um código. ..... 26
Figura 3.1: Estrutura proposta para realização do experimento. ..... 29
Figura 3.2: Bancada montada no laboratório para auxiliar na programação. ..... 29
Figura 3.3: Tela principal do ambiente de desenvolvimento. ..... 30
Figura 3.4: Tela do software *Virtual Serial Ports Emulator*. ..... 31
Figura 3.5: Tela principal do software 232 Analyzer. ..... 31
Figura 3.6: Tela principal do software DeviceMonitoring Studio. ..... 32
Figura 3.7: Função responsável pela criação da conexão com a porta serial. ..... 32
Figura 3.8: Função de validação de criação da comunicação com a porta serial. ..... 33
Figura 3.9. Fluxograma para a criação da conexão ..... 34
Figura 3.10: Configuração DCB utilizada para definir os parâmetros de comunicação. ..... 34
Figura 3.11: Fluxograma para a configuração da transmissão (RS-232). ..... 35
Figura 3.12: Função responsável pelo encerramento da conexão. ..... 36
Figura 3.13: Código de telegrama de leitura. ..... 37
Figura 3.14: Fluxograma de leitura de dados. ..... 37
Figura 3.15: Código do telegrama de escrita. ..... 38
Figura 3.16: Fluxograma de escrita de dados. ..... 39
Figura 3.17: Função de conversão de Hexadecimal para decimal. ..... 40
Figura 3.18: Função de conversão Hexadecimal para Decimal. ..... 41
Figura 3.19: Diagrama de estados de uma porta XOR. ..... 42
Figura 3.20: Função de geração do BCC. ..... 43
Figura 4.1: Primeira tela para monitoramento. ..... 44
Figura 4.2: Segunda tela de monitoramento. ..... 45
Figura 4.3: Software *Device Monitoring Studio* monitorando o tráfego na porta serial. ..... 46
Figura 4.4: Detalhamento do monitoramento de escrita e leitura de dados. ..... 46
Figura 4.5: Exemplo de leitura e resposta via comunicação serial. ..... 46
Figura 4.6: Retorno da leitura lida convertido em decimal mostrado no campo com índice 333. ... 47
Figura 4.7: Interface principal com o software funcionando. ..... 48
Figura 4.8. Tela de gráficos com software funcionando. ..... 48

# Lista de Tabelas

Tabela 3.1: Possibilidades de configuração de transmissão. ..... 36
Tabela 3.2: Exemplo de conversão de hexadecimal para decimal. ..... 41
Tabela 3.3: Exemplo de conversão de decimal para hexadecimal. ..... 42

# Lista de Símbolos

IHM: Interface Homem-Máquina

S.O.: Sistema Operacional

CV: Cavalo Vapor

V: *Volts*

A: *Ampere*

API: *Application Programming Interface*

RPM: Rotação Por Minuto

GND: *Ground*

PID: Proporcional, Integrativo, Derivativo

OEM: *Original Equipment Manufacturers*

# SUMÁRIO


1 INTRODUÇÃO ..... 12

1.1 Contextualização ..... 12

1.2 Justificativa ..... 12

1.3 Objetivos ..... 13

1.3.1 Objetivo Geral ..... 13

1.3.2 Objetivos Específicos ..... 13

1.4 Estrutura do trabalho ..... 14

2 REVISÃO DA LITERATURA ..... 15

2.1 Introdução ..... 15

2.2 Sistemas operacionais ..... 15

2.2.1 Windows ..... 15

2.2.2 Linux ..... 16

2.2.3 Mac OS ..... 16

2.2.4 Recursos para protocolo de comunicação ..... 17

2.3 Softwares supervisórios ..... 18

2.3.1 Elipse SCADA ..... 18

2.3.2 RS View ..... 19

2.3.3 WinCC ..... 19

2.4 Linguagens de programação para desenvolvimento de software supervisório ..... 20

2.5 Inversor CFW-09 ..... 20

2.6 Interface de comunicação ..... 21

2.6.1 Descrição das interfaces ..... 21

2.6.2 Interface RS-232 ..... 22

2.6.3 Kit de comunicação RS-232 para PC ..... 22

2.6.4 Protocolo de comunicação ..... 23

2.6.4.1 Telegrama de leitura ..... 23

2.6.4.2 Telegrama de Escrita ..... 24

2.6.4.3 Execução e Teste de Telegrama ..... 25

2.6.4.3.1 Telegrama de leitura: ..... 26

2.6.4.3.2 Telegrama de escrita: ..... 26

2.6.4.4 Códigos de Variáveis ..... 26

3 DESENVOLVIMENTO DO SOFTWARE E RESULTADOS ..... 28

3.1 Materiais e métodos ..... 28
3.1.1 Hardware ..... 28
3.1.2 Software ..... 29
3.2 Conexão: Mestre/Escravo ..... 32
3.2.1 Estabelecendo conexão ..... 32
3.2.2 Processo de leitura ..... 36
3.2.3 Processo de escrita ..... 38
3.3 Funções auxiliares ..... 39
3.3.1 Conversor hexadecimal para decimal ..... 40
3.3.2 Conversor decimal para hexadecimal ..... 41
3.3.3 Gerador BCC ..... 42
4 RESULTADOS E ANÁLISES ..... 44
4.1 Interfaces ..... 44
4.2 Testes ..... 45
4.3 Validação do sistema ..... 47
4.4 Documentação ..... 49
5 CONCLUSÕES ..... 50
5.1 Considerações Finais ..... 50
5.2 Trabalhos Futuros ..... 50
5.3 Publicação ..... 51
REFERÊNCIAS BIBLIOGRÁFICAS ..... 52
APÊNDICE A – MANUAL DO USUÁRIO ..... 54
APÊNDICE B - CÓDIGO FONTE ..... 59
APÊNDICE C – ARTIGO ACEITO PELA ICECE 2011 ..... 73

# 1 INTRODUÇÃO

## 1.1 CONTEXTUALIZAÇÃO

Nos últimos anos, com o rápido desenvolvimento das tecnologias na informática e na transmissão de dados, o controle dos sistemas de automação em escala industrial vem adquirindo uma importância crescente a partir de peças industriais e acadêmicas, e das tecnologias computacionais que tem sido amplamente aplicada para o sistema de monitoramento, facilitando a supervisão e controle do processo de produção, aumentando a confiabilidade do sistema [1].

Para um sistema de informações de uma indústria ser bem sucedido, alguns pontos devem ser devidamente observados, tais como a aquisição de dados necessários para o cálculo dos índices de uma produção para isto deve ser realizado um processamento de dados com precisão e em tempo real devem ser feitos continuamente e o armazenamento de dados para a recuperação da informação é obrigatório [2].

## 1.2 JUSTIFICATIVA

No monitoramento de uma planta industrial pode-se notar a diferença entre um sistema usando uma interface homem máquina ou um software supervisório. A automação industrial utilizando um software supervisório apresenta diversos recursos que as Interfaces Homem-Máquina (IHMs) não possuem, como por exemplo, a geração de relatório, um banco de dados para armazenamento das informações além de maiores capacidades de processamento [3].

Além da facilidade de relatórios e confiabilidade dos dados obtidos os softwares supervisórios podem ser instalados em locais diferentes de onde se encontra o dispositivo controlado, podendo-se assim controlar toda uma planta de uma indústria apenas em um local desde que este contenha canais de comunicação com todos os equipamentos que serão monitorados e/ou controlados.

Assim é essencial que as escolas de formação de áreas correlatas a automação industrial se adéquem e treinem nestes tipos de aplicações industriais.

Algumas empresas já fornecem software de monitoramento e atuação, contudo geralmente são softwares com funções específicas deixando assim o usuário restrito apenas as funcionalidades disponibilizadas pelo fabricante.

Deste modo, foi proposto nesta dissertação o desenvolvimento de um software capaz de monitorar, atuar e controlar equipamentos industriais, com uma interface amigável e de fácil entendimento pelos usuários e disponibilizar aos futuros alunos uma programação que permita a criação de códigos de controle.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo Geral

O objetivo geral desta dissertação é criar um novo software para monitoramento e supervisão de equipamentos industriais, mas com uma interface amigável para facilitar o entendimento dos alunos quando estudarem esta área de conhecimento.

### 1.3.2 Objetivos Específicos

Para atingir este objetivo geral foi necessário obedecer a objetivos específicos, estes detalhados abaixo:

- Estudar softwares existentes para computadores que permitam realizar o monitoramento e a atuação em equipamentos industriais;
- Estudar as linguagens de programação e ferramentas de desenvolvimento de softwares disponíveis e selecionar a mais adequada para o desenvolvimento do projeto;
- Estudar o conversor de frequência, comercialmente denominado inversor de freqüência, modelo CFW-09, fabricado pela Weg Automação, equipamento industrial utilizado para testes a fim de validar o software desenvolvido;
- Estudar os conceitos de sistemas de supervisão;

- Efetuar a modelagem e posterior implementação das rotinas para construção do software para monitoramento e atuação;
- Programar e testar o software de monitoramento e atuação para o inversor CFW-09;

## 1.4 ESTRUTURA DO TRABALHO

Neste capítulo foi feita uma introdução ao trabalho proposto, depois descrito a justificativa mostrado o porquê do trabalho, e por fim os objetivos propostos pelo trabalho.

No capítulo 02 é descrito a revisão de literatura abordado no trabalho, é feito uma descrição dos sistemas operacionais, depois é abordado alguns dos softwares supervisórios existentes, a seguir é mostrado algumas linguagens de programação existentes e por fim é descrito como funciona o conversor de freqüência CFW-09.

No capítulo 03 é descrito como foi desenvolvido o software, mostrando os materiais e métodos adotados para criação do protótipo, também é descrito algumas funções auxiliares que foram utilizadas para obtenção dos resultados.

No capítulo 04 são descritos os resultados obtidos com base nas interfaces descritas e com testes de validação do sistema alem da descrição da documentação feita para utilização do software desenvolvido.

No capítulo 05 é descrito as considerações finais do trabalho, propostas de continuidades do projeto e é mostrado as publicações feitas com base neste trabalho.

# 2 REVISÃO DA LITERATURA

## 2.1 INTRODUÇÃO

No desenvolvimento de um software supervisório o projetista devesse atentar em qual plataforma o produto terá o melhor desempenho e os recursos computacionais que serão disponibilizados. Neste capítulo descreve-se sobre os principais sistemas operacionais e sobre alguns softwares supervisórios que possuem uma correlação com este trabalho.

## 2.2 SISTEMAS OPERACIONAIS

Um sistema operacional deve gerenciar os componentes físicos de um computador e fornecer aos programas do usuário uma interface com o hardware mais simples [4].

Atualmente, existem três sistemas operacionais comuns em computadores de mesa (*Desktop*): o Windows, o Mac e o Linux [5]. Cada um com sua característica como descrito a seguir.

### 2.2.1 Windows

A família de sistemas operacionais Windows, foi criado pela Microsoft empresa fundada por Bill Gates e Paul Allen.em 1981, possui diversas versões as mais conhecidas são o Windows Xp e o Windows 98 versões lançadas em 2001 e 1998 respectivamente [3].

O Windows se tornou popular devido a uma estratégia de marketing adotada pela Microsoft, que fechando parcerias com diversos fabricantes de computadores inseriu o seu sistema operacional nas máquinas novas assim os usuários que adquirissem computadores novos já recebiam com uma versão do Windows instalado.

O Windows Seven é a versão mais atual de sistema operacional disponibilizado pela empresa lançado em 2009 para usuários comuns e para servidores possui uma versão específica, o Windows Server 2008 R2 também lançado em 2009.

A Microsoft desenvolvedora desse sistema operacional possui todos os direitos sobre seu código fonte e isso faz com que alterações possam ser feitas apenas pelos

programadores da empresa, diferente do Linux que possui uma licença GPL onde qualquer pessoa pode modificar seu código fonte.

### 2.2.2 Linux

O nome Linux é usado para todo sistema operacional que tenha como núcleo o Linux, este núcleo foi desenvolvido por Linus Torvalds baseado no Unix. Sua licença é GPL permitindo que qualquer pessoa tenha acesso ao seu código fonte e o altere quando desejado [5].

O Linux tem destacado-se como principal sistema operacional para servidores de grande porte devido a sua estabilidade, porém não muito utilizado por usuários comuns devido a sua aparência não muito “amigável” [3].

Com o passar do tempo foram sendo desenvolvidas novas interfaces para o Linux deixando o ambiente mais amigável uma característica que o Windows tinha desde sua primeira interface gráfica [6].

Grandes companhias como a Red Hat, a Suse, a Mandriva e a Canonical desenvolvem distribuições baseados no Linux além de comunidades no mundo inteiro que ajudam a desenvolver o sistema operacional [6].

### 2.2.3 Mac OS

O Mac OS é o sistema operacional padrão dos computadores Macintosh ou também chamados de Mac, com algumas características do Windows, este ambiente foi desenvolvido por Jef Raskin em 1984 e tinha como objetivo de popularizar as interfaces gráficas e acabou sendo muito utilizado no tratamento de vídeo, imagem e som [7].

Assim como o Windows o código é fechado, podendo apenas os programadores da Apple (empresa criada por Jef Raskin) e é vendido em como sistema operacional nativo nas maquinas vendidas pela sua empresa os Macintosh [7].

O Mac OS tem uma interface gráfica muito amigável e é muito utilizado para tratamento de imagens, mas se torna uma máquina não muito útil para usuários comuns devido ao seu alto custo e não possuir uma grande quantidade de softwares disponíveis assim como no Windows ou no Linux.

O Windows é um sistema operacional popular na indústria e em computadores domésticos. O Linux é um sistema operacional flexível, porém requer alto conhecimento dos seus usuários. O Mac OS vem sendo utilizado principalmente em máquinas específicas principalmente na área de edição gráfica devido as suas características próprias desenvolvidas pela Apple Macintosh [5][6][7].

### 2.2.4 Recursos para protocolo de comunicação

Para o acesso direto a porta serial se faz necessário o conhecimento das API's (*Application Programming Interface* - Interface de Programação de Aplicações) do Windows, estas API's são pequenos programas dentro do sistema operacional que auxiliam na execução de várias atividades dentro do sistema operacional [8].

A Microsoft, desenvolvedora do Windows disponibiliza na internet as bibliotecas disponíveis assim como exemplos de configurações e uso, facilitando a comunicação entre os programas desenvolvidos por usuários e o sistema operacional.

Algumas funções prontas foram utilizadas para criar uma conexão entre o software e o sistema operacional podendo se destacar as funções:

- CreateFile ( ): Função responsável pela abertura de um canal de comunicação entre o computador e a porta serial liberando assim suas funções perante ao S.O.
- CloseHandle ( ): Função responsável pelo fechamento da conexão entre o computador e a porta serial do S.O.
- BuildCommDCB ( ): Função responsável pelo preenchimento dos campos da estrutura DCB com valores especificados numa string de controle de dispositivos.
- ReadFile ( ): Função de leitura de dados via porta serial, esta função é utilizada toda vez que precisa-se ler algo que esteja no barramento serial.
- WriteFile ( ): Função de escrita de dados via porta serial, esta função é utilizada toda vez que precisa-se escrever algo no barramento serial.
- GetCommState ( ) // SetCommState ( ): Estas funções respectivamente tratam de ler e alterar informações referentes ao parâmetro de comunicação serial como taxa de transmissão, paridade, entre outros.

- GetCommTimeouts( ) // SetCommTimeouts( ): Estas funções são responsáveis respectivamente por ler e alterar as informações de *Timeouts* dependendo do protocolo utilizado pode-se alterar o tempo de espera de leitura de um bit na porta serial ou verificar se foi passado este tempo.

## 2.3 SOFTWARES SUPERVISÓRIOS

Os sistemas supervisórios podem ser considerados como o nível mais alto de IHM pois mostram o que esta acontecendo no processo e permitem ainda que se atue neste.

O software supervisório é visto como o conjunto de programas gerados e configurado no software básico de supervisão, implementando as estratégias de controle e supervisão com telas gráficas de interfaceamento homem-máquina que facilitam a visualização do contexto atual, a aquisição e tratamento de dados do processo e a gerencia de relatórios e alarmes.

No mercado existem vários ambientes de desenvolvimentos para softwares supervisórios, dentre eles o Elipse SCADA da Elipse Software, o RS View da Hockwell e o WinCC da Siemens cada um com sua característica descritas a seguir.

### 2.3.1 Elipse SCADA

Segundo seu fabricante, a Elipse Software, podem-se tomar como características do software [9]:

Vantagens:

- Diversos recursos que facilitam e agilizam o desenvolvimento de uma aplicação;
- Totalmente configurável pelo usuário;
- Permite a monitoração de variáveis em tempo real, através de gráficos e objetos que estão relacionados com as variáveis físicas de campo;
- É possível fazer acionamentos e enviar e receber informações para equipamentos de aquisição de dados.

Desvantagens:

- Linguagem de programação própria chamada de Elipse Basic;
- Disponibilizado apenas em plataforma Windows;

- Software pago.

### 2.3.2 RS View

De acordo com a Hockwell as principais características são [10]:

Vantagens:

- Pode ser programado em Visual Basic;
- Possui suporte as mais populares aplicações industriais desenvolvidos por OEM's.

Desvantagens:

- Restrito apenas as bibliotecas prontas de comunicação, restringindo assim o seu uso;
- Disponibilizado apenas em plataforma Windows;
- Software pago;

### 2.3.3 WinCC

A Siemens caracteriza seu software como [11]:

Vantagens:

- Simples configuração do projeto;
- Facilidade na manutenção de projetos;
- Facilidade na criação de IHM's devido a ser este seu principal foco.

Desvantagens:

- Requer alto conhecimento do software, devido a sua linguagem de programação própria;
- Disponibilizado apenas em plataforma Windows;
- Software pago;

Pode-se notar que cada fabricante possui suas características, fazendo assim com que cada software se adéque as suas especificações; como linguagens de programação própria ou a plataforma a ser utilizada, concluindo, os softwares disponibilizados são pagos encarecendo assim os custos de projeto.

## 2.4 LINGUAGENS DE PROGRAMAÇÃO PARA DESENVOLVIMENTO DE SOFTWARE SUPERVISÓRIO

Para a criação de um sistema de monitoramento existem várias linguagens de programação no mercado, com isto foi necessário um estudo prévio e comparação entre algumas mais conhecidas, a fim de escolher a que melhor se adequasse ao projeto proposto.

Para análise foram utilizadas as linguagens JAVA e C++ sendo feita uma descrição de cada.

A linguagem Java se tornou a linguagem popular para programar aplicativos baseados em Intranet e Internet e softwares para dispositivos que se comunicam através de uma rede, outro fator atrativo é que a tecnologia JAVA é totalmente orientada a objetos [12].

A linguagem de programação C++ foi desenvolvida para propósito geral com suporte para diversas aplicações [13].

Uma quantidade enorme de código C foi escrita nas empresas nas últimas décadas. O C++ por ser uma evolução da linguagem C, os programadores podem executar código C, compilá-lo em um compilador C++, freqüentemente com alterações apenas na nomenclatura das palavras reservadas [12].

Ambas as tecnologias possuem uma grande quantidade de documentação disponível na internet assim como livros, além de diversos ambientes de desenvolvimentos que auxiliam na programação.

O que levou a escolha do C++ como linguagem a ser utilizada foi à facilidade de integração entre os protocolos de comunicação com o sistema operacional já que o Windows possui bibliotecas escritas em C facilitando assim todo processo de desenvolvimento do sistema.

## 2.5 INVERSOR CFW-09

De acordo com o manual do usuário do fabricante [14], este dispositivo é destinado para o acionamento de motores trifásicos com potência de 1,5 a 1500 cv. Este inversor tem incorporado as mais avançadas tecnologias disponíveis para variação de velocidade em motores de indução trifásicos.

As principais aplicações do inversor se dão principalmente em bombas centrífugas, bombas dosadoras de processo, ventiladores, exaustores, agitadores, misturadores, extrusoras, esteiras transportadoras, mesas de rolos, granuladores, secadores, fornos rotativos, máquinas de corte e solda, elevadores, sistemas multibombas, etc.

Entre as diversas características encontradas no inversor podem-se citar algumas que ajudaram na escolha deste modelo para desenvolvimento do projeto:

- Potência: De 1,5 a 1500 cv tornando-se um modelo com abrangência de aplicações capaz de se adaptar a novas necessidades.
- Controle: Escalar e vetorial embutido criando possibilidades de comparação entre os tipos de controle alem de um controle PID integrado.
- Comunicação: Possui a possibilidade de comunicação entre outros dispositivos via RS – 232 ou RS – 485, facilitando o acesso às informações;
- Configuração: Disponibilidade de 100 parâmetros configuráveis pelo usuário via software (fornecido pela fabricante separadamente) ou IHM facilitando o acesso às informações.

## 2.6 INTERFACE DE COMUNICAÇÃO

Nesta seção é descrito como utilizar o kit de comunicação serial, disponível como um dispositivo opcional pela WEG, aborda-se também o protocolo de comunicação entre o inversor e o mestre e por fim demonstram-se alguns exemplos de telegramas de leitura e de escrita.

### 2.6.1 Descrição das interfaces

Existem duas interfaces possíveis para ligação entre o inversor e o mestre, a RS-232 a qual é utilizada ponto a ponto e até 10 metros, e a RS-485, que é multiponto com isolamento galvânico em até 100 metros. Devido à popularidade da interface RS-232, neste trabalho adotou-se esta interface para realizar este trabalho.

### 2.6.2 Interface RS-232

Utilizando a interface RS-232 a ligação direta entre um mestre e inversor, ou seja, ponto a ponto, os dados podem ser transmitidos de forma bidirecional, porém não simultânea chamada de *HALF DUPLEX*. São utilizados três fios para comunicação: o (TX) para transmissão, (RX) para recepção; e (GND) para retorno;

### 2.6.3 Kit de comunicação RS-232 para PC

É possível fazer a supervisão do CFW-09 da WEG utilizando a interface serial RS-232, o protocolo é baseado no tipo pergunta/resposta conforme as normas regulamentadores ISO 1745 e ISO 646 utilizando caracteres do tipo ASCII entre inversor e mestre. Este mestre pode ser um computador, um controlador lógico programável (CLP) entre outros. A velocidade máxima de transmissão é de 9600 bits por segundo.

Para se utilizar esta interface deve-se fazer uso do módulo RS-232 vendido separadamente pela WEG, este módulo substitui a IHM e disponibiliza uma comunicação via RS-232 utilizando o conector RJ11. Caso ainda seja necessária a utilização da IHM, o módulo RS-232 também provê a conexão para a mesma. Na Figura 2.1 mostra-se o módulo de comunicação RS-232.

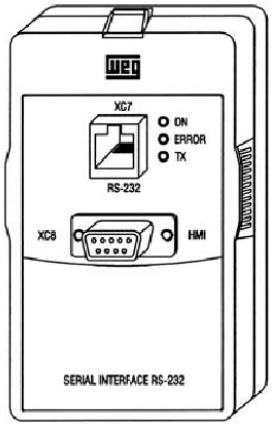


Figura 2.1: Módulo de comunicação RS-232

O Kit de Comunicação RS-232 para PC permite a conexão do CFW-09 a um PC através da interface RS-232 e é composto de um módulo RS-232 Serial Interface, cabo 3

metros RJ–11 para DB9 e o software SuperDrive para Windows que permite a programação, operação e monitoração do CFW-09.

Para instalação do kit de comunicação RS-232 deve-se retirar a IHM do inversor, instalar o módulo RS-232 no lugar da IHM, instalar o software SuperDrive (software desenvolvido pela WEG para comunicação entre o computador e o inversor) no computador e por fim conectar o PC ao inversor usando o cabo conector.

### 2.6.4 Protocolo de comunicação

O protocolo de transmissão segue a norma ISO 1745, esta norma tem como base o uso de caracteres de controle reservados para "funções de controle de transmissão" em ASCII. São usadas também somente seqüências de caracteres de texto sem cabeçalho. A monitoração dos erros é feita através de transmissão relacionada à paridade dos caracteres individuais, e pela ISO 646 esta norma define o conjunto universal de caracteres definindo apenas uma codificação para cada caractere. A monitoração de paridade é feita conforme DIN 66219 (paridade par).

São usados dois tipos de mensagens (pelo mestre):

- Telegrama de leitura: para consulta do conteúdo das variáveis dos inversores;
- Telegrama de escrita: para alterar o conteúdo das variáveis ou enviar comandos para os inversores.

#### 2.6.4.1 Telegrama de leitura

Este telegrama permite que o mestre (neste caso o computador) receba do inversor o conteúdo da variável pedida. No telegrama de resposta o inversor transmite os dados solicitados pelo mestre. Na Figura 2.2 pode-se verificar este exemplo de telegrama.

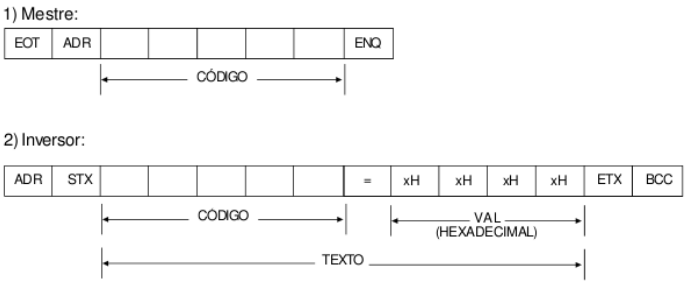


Figura 2.2: Exemplo de telegrama de leitura.

Formato do telegrama de leitura:

- EOT: caractere de controle EndOfTransmission;
- ADR: endereço do inversor (ASCII@, A, B, C, a ) (ADdRess);
- CÓDIGO: endereço da variável de 5 dígitos codificados em ASCII;
- ENQ: caracter de controle ENQuiry (solicitação).

Formato do telegrama de resposta do inversor:

- ADR: caracter - endereço do inversor;
- STX: caracter de controle - Start ofTeXt;
- TEXTO: consiste em:
- CÓDIGO: endereço da variável;
- " = ": caracter da separação;
- VAL: valor em 4 dígitos HEXADECIMAIS;
- ETX: caracter de controle - EndofTeXt;
- BCC: Byte de CheCksum - EXCLUSIVE OR de todos os bytes entre STX (excluído) e ETX (incluído).

2.6.4.2 Telegrama de Escrita

Este telegrama envia dados para as variáveis dos inversores. O inversor irá responder indicando se os dados foram aceitos ou não conforme mostrado na Figura 2.3.

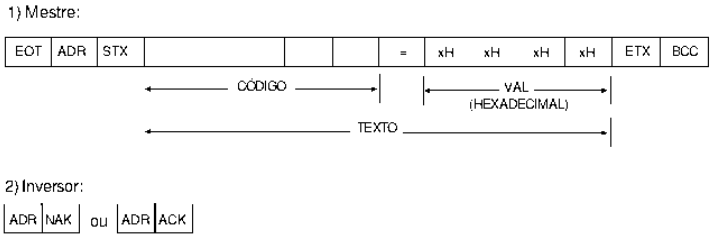


Figura 2.3: Exemplo de telegrama de escrita.

Formato do telegrama de escrita:

- EOT: caracter de controle EndOfTransmission;
- ADR: endereço do inversor;
- STX: caracter de controle Start ofTeXt;
- TEXTO: consiste em:
- CÓDIGO: endereço da variável;
- " = ": caracter de separação;
- VAL: valor composto de 4 dígitos HEXADECIMAIS;
- ETX: caracter de controle EndofTeXt;
- BCC: Byte de CheCksum - EXCLUSIVE OR de todos os bytes entre STX (excluído) e ETX (incluído).

Formato do telegrama de resposta do inversor:

*Aceitação:*

- ADR: endereço do inversor;
- ACK: caracter de controle ACKnowledge;

*Não aceitação:*

- ADR: endereço do inversor;
- NAK: caracter de controle NotAcKnowledge.

### 2.6.4.3 Execução e Teste de Telegrama

Os inversores e o mestre testam a sintaxe do telegrama. A seguir são definidas as respostas para as respectivas condições encontradas:

*2.6.4.3.1 Telegrama de leitura:*

Quando se faz a leitura de algum parâmetro no inversor, podem-se obter três resultados:

Sem resposta: com estrutura do telegrama errada, caracteres de controle recebidos errados ou endereço do inversor errado;

NAK: CÓDIGO correspondente à variável inexistente ou variável só de escrita;

TEXTO: com telegramas válidos.

*2.6.4.3.2 Telegrama de escrita:*

Quando se faz a escrita de algum parâmetro no inversor, podem-se obter três resultados:

Sem resposta: com estrutura do telegrama errada, caracteres de controle recebidos errados ou endereço do inversor errado;

NAK: com código correspondente à variável inexistente, BCC (byte de checksum) errado, variável só de leitura, VAL fora da faixa permitida para a variável em questão, parâmetro de operação fora do modo de alteração destes;

ACK: com telegramas válidos;

2.6.4.4 Códigos de Variáveis

O campo denominado de código contém o endereço de parâmetros e variáveis básicas composto de 5 dígitos (caracteres ASCII) conforme apresentado na Figura 2.4.:

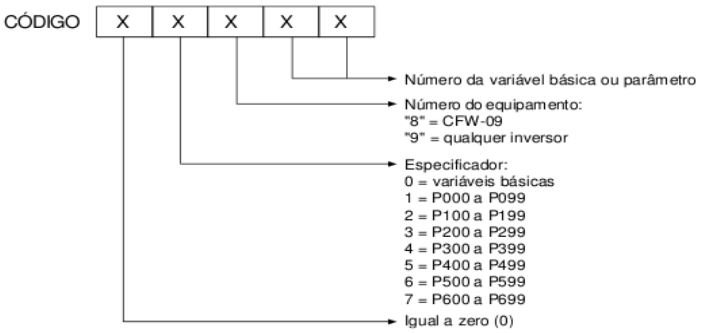


Figura 2.4: Combinação de campos para criação de um código.

Com base na Figura 2.4, para envio de um código referente ao parâmetro 004 é necessário o envio do código 0 1 8 04 onde 0 é definido por padrão, 1 refere-se ao grupo de parâmetros P000 a P099, 8 indica o modelo do inversor a ser enviado e 04 o complemento número a ser acessado identificando assim o parâmetro 004.

# 3 DESENVOLVIMENTO DO SOFTWARE E RESULTADOS

Este capítulo descreve as etapas de desenvolvimento do sistema capaz de supervisionar a planta adotada. Este capítulo também aborda os paradigmas de programação para resolução de alguns problemas enfrentados durante o desenvolvimento do trabalho.

Para isto o capítulo foi dividido em três seções: os materiais e métodos utilizados para auxiliar na conclusão do trabalho; os passos para conexão entre o supervisório e a planta; por fim são descritas algumas funções auxiliares criadas para auxiliar no entendimento e na execução do programa.

## 3.1 MATERIAIS E MÉTODOS

Neste tópico descrevem-se os materiais e métodos utilizados para a criação do software proposto, este se dividiu em duas etapas. Na primeira realiza-se a descrição do hardware utilizado e na segunda aborda-se a descrição do software empregado para elaboração do sistema.

### 3.1.1 Hardware

Para elaboração do software foi necessário que uma estrutura fosse montada para realização dos testes práticos. O diagrama com a estrutura proposta para realização do projeto é ilustrado na Figura 3.1. A estrutura é composta por um computador (A), um conversor de frequência (B), um motor de indução (C), um gerador de corrente contínua (D), um banco de lâmpadas (E) funcionando como uma carga variável e um circuito auxiliar foi utilizado para regular a tensão descrito em (F).

Na Figura 3.2 pode-se observar a bancada montada no laboratório para a programação e testes. O gerador de corrente contínua com excitação independente (variável) alimentando uma carga variável (quadro de lâmpadas) foi usado para possibilitar a variação de carga no eixo do motor de indução, possibilitando dessa forma a variação da potência suprida pelo conversor de frequência.

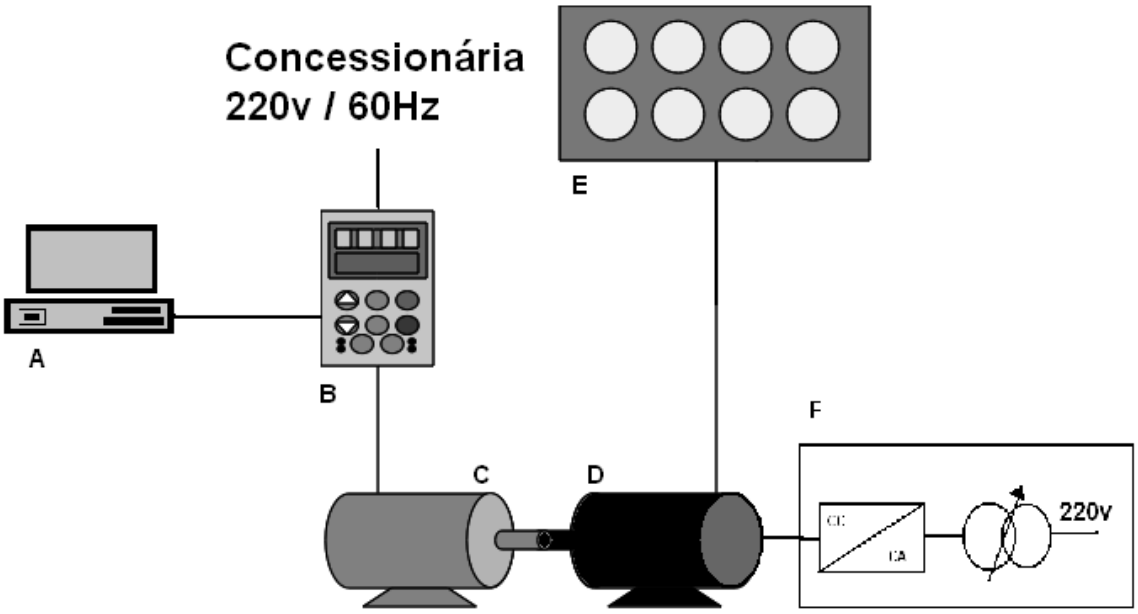


Figura 3.1: Estrutura proposta para realização do experimento.

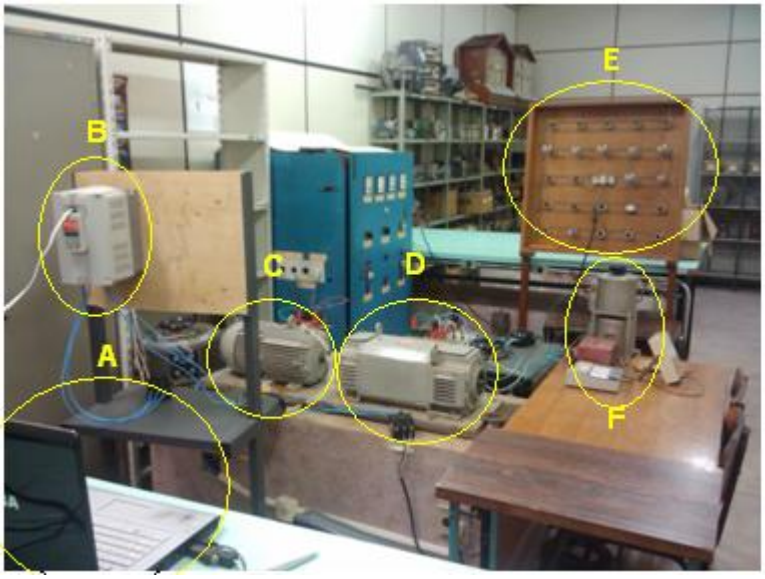


Figura 3.2: Bancada montada no laboratório para auxiliar na programação.

### 3.1.2 Software

Definida a linguagem de programação a ser utilizada, foi escolhido o software da Embarcadero Tecnologies® chamado de CodeGear™ C++ Builder® 2009. Este se apresentou mais completo e com uma versão de demonstração, mais indicada para o processo de programação devido a sua vasta abrangência e não gerando nenhum custo com licenças [15].

Na Figura 3.3 é mostrado à tela principal do ambiente de desenvolvimento utilizado para desenvolvimento do software.

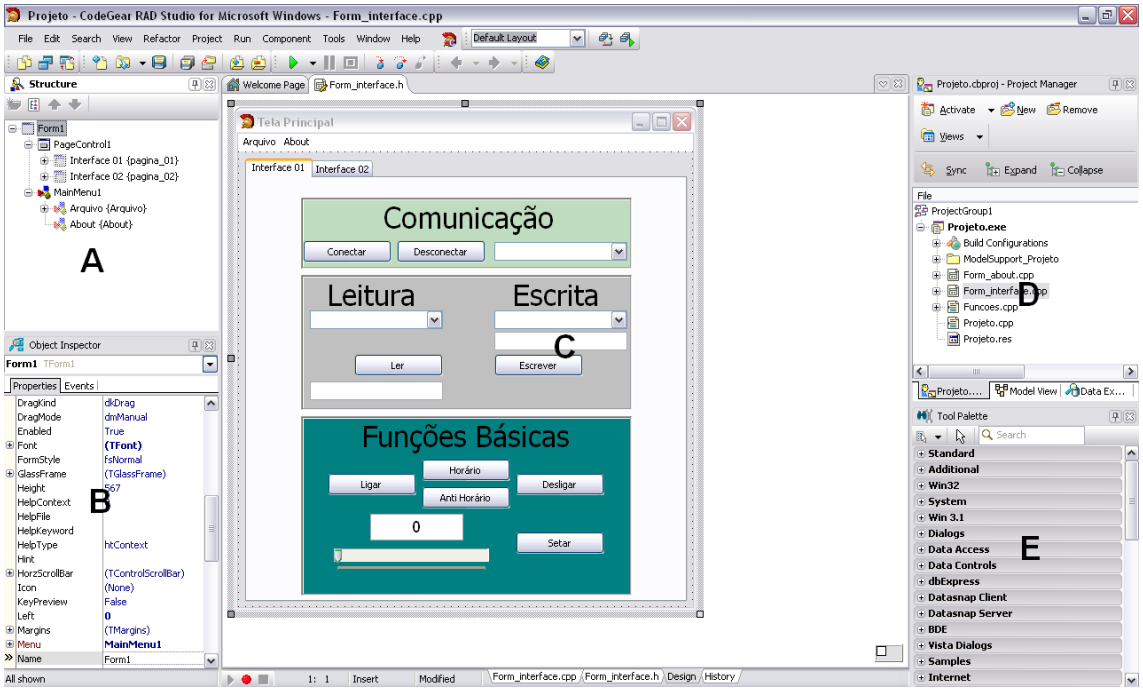


Figura 3.3: Tela principal do ambiente de desenvolvimento.

Na Figura 3.3 (A) é mostrada a estrutura da interface do software em desenvolvimento, em (B) é o inspetor de objetos, com este menu pode-se verificar quais alterações podem ser feitas em cada objeto, (C) representa a tela principal, nesta tela é feita toda programação tanto visual (com botões, menus, etc) quanto a programação em modo texto, (D) mostra a estrutura dos arquivos gerados pelo programa e (E) é mostrado a paleta de ferramentas, essa paleta possui diversos itens que podem ser inseridos como botões ou menus interativos.

Alguns softwares foram utilizados para auxiliar no desenvolvimento da programação. Na Figura 3.4 é mostrada a tela do software *Virtual Serial Ports Emulator*, um software *freeware* com capacidade de criar portas seriais virtuais em um computador, este software foi utilizado para simular uma porta serial [16].

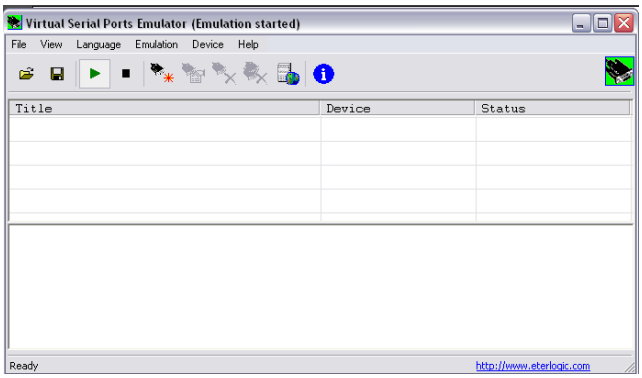


Figura 3.4: Tela do software *Virtual Serial Ports Emulator*.

O Programa 232 Analyzer mostrado na Figura 3.5, foi utilizado para simulação de envio e recebimento de pacotes onde este software simulou um conversor de freqüência comunicando-se com o sistema desenvolvido [17].

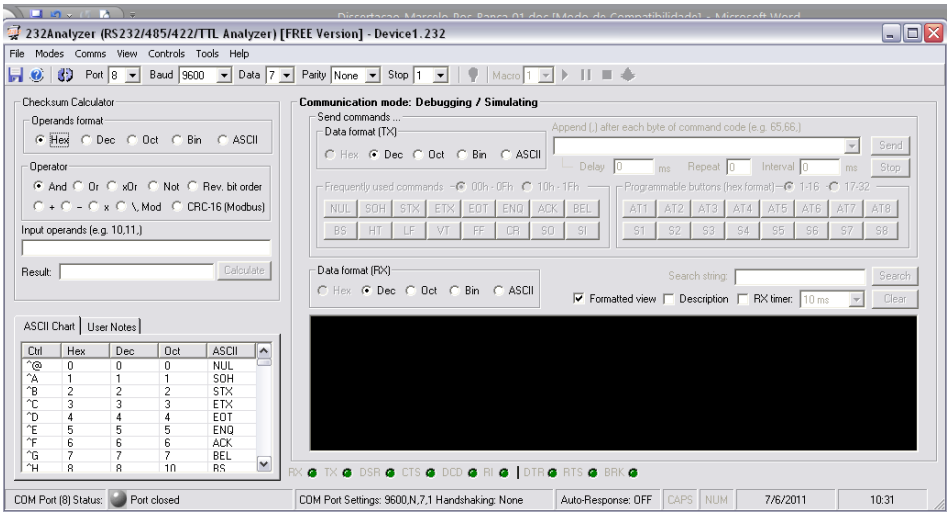


Figura 3.5: Tela principal do software 232 Analyzer.

Na Figura 3.6 é mostrada a tela do software Device Monitoring Studio um programa capaz de monitorar todos os dados enviados e recebidos pela porta serial, este programa foi usado com o intuito de se verificar a autenticidade dos dados enviados e recebidos pela serial [18].

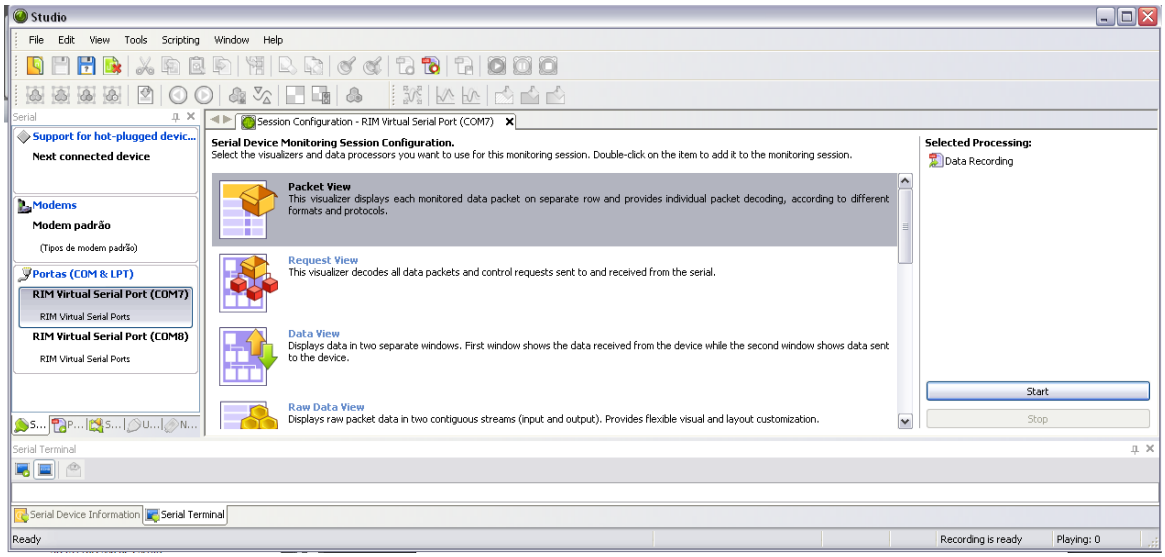


Figura 3.6: Tela principal do software DeviceMonitoring Studio.

## 3.2 CONEXÃO: MESTRE/ESCRAVO

### 3.2.1 Estabelecendo conexão

Para programação do protocolo foi necessário o conhecimento de algumas bibliotecas dentro do ambiente de desenvolvimento utilizado, dentre elas pode-se citar a biblioteca chamada *Windows.h.* Esta biblioteca é responsável por todos os parâmetros que podem ser ligados a este sistema operacional. Neste caso esta biblioteca foi utilizada para se ter acesso aos comandos descritos na seção 2.3.2.

A partir desta biblioteca podem-se utilizar os parâmetros de criação e configuração da porta serial. Os parâmetros para uma nova conexão são demonstrados na Figura 3.7.

```
80  hCom = CreateFile    (
                          NomePorta,    //Nome da porta.
                          GENERIC_READ | GENERIC_WRITE,    //Para leitura e escrita.
                          0,        //Nenhuma outra abertura será permitida.
                          NULL,  //Atributos de segurança. (NULL) padrão.
                          OPEN_EXISTING,  //Criação ou abertura.
                          0, //Entrada e saída sem overlapped.
                          NULL      //Atributos e Flags. Deve ser NULL para COM.
                          );
```

Figura 3.7: Função responsável pela criação da conexão com a porta serial.

Na Figura 3.8, tem-se a hCom nome dado a nova conexão, CreateFile significa que um novo arquivo será criado, NomePorta é uma variável que carrega o endereço da COM que será utilizada, GENERIC_READ | GENERIC_WRITE são responsáveis para habilitar a leitura e a escrita da comunicação, 0 (zero) determina que nenhuma outra abertura de comunicação com esta porta será permitida, NULL (nulo) os atributos de segurança não estão conhecidos, OPEN_EXISTING esta faz com que a porta seja aberta ou crie uma nova abertura de porta, 0 (zero) este segundo zero é utilizado para determinar a não existência de overlapped e por fim NULL informando que os atributos e Flags devem ser nulos para a porta COM.

Após esta criação é feito um uma comparação para validar a criação da conexão, comparando-se a variável hCom com o valor “INVALID_HANDLE_VALUE” se a função hCom for igual a esta definição significa que não foi possível abrir a porta especificada, esta comparação é mostrada na Figura 3.8.

```
90      if(hCom == INVALID_HANDLE_VALUE)
        {
              return 1;    //Erro ao tentar abrir a porta especificada.
        }
```

Figura 3.8: Função de validação de criação da comunicação com a porta serial.

O *return* 1 (retorno um) significa que a função retorna 1 e no teste de validação este retorno significa que houve erro na criação da comunicação, na Figura 3.9 é mostrado o fluxograma da criação e validação de uma nova conexão.

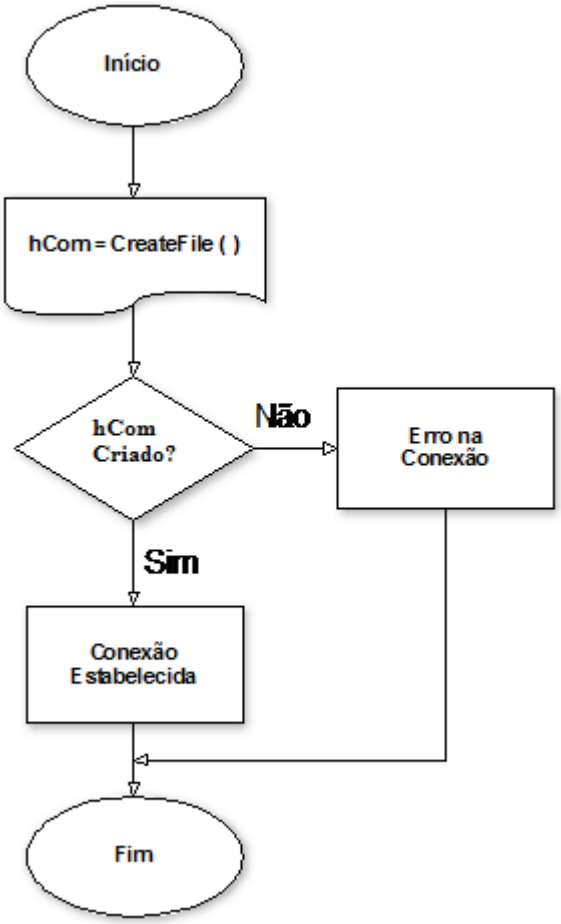


Figura 3.9. Fluxograma para a criação da conexão

Depois de concluída esta etapa de criação de uma nova conexão, é necessária a configuração dos parâmetros como velocidade de transmissão, quantidade de caracteres, paridade e quantidade de bits de parada, para isto foi utilizada a função descrita na Figura 3.10.

```
DCB dcb; //Estrutura DCB é utilizada para definir todos os parâmetros da comunicação.
dcb.BaudRate = CBR_9600;
dcb.ByteSize = 7;
dcb.Parity = EVENPARITY;
dcb.StopBits = ONESTOPBIT;

//Define novo estado.
if( SetCommState(hCom, &dcb) == 0 )
{
    return 1;   //Erro.
}
```

Figura 3.10: Configuração DCB utilizada para definir os parâmetros de comunicação.

Primeiramente tem-se o DCB dcb que é uma estrutura utilizada para definir os parâmetros da comunicação. Após tem-se as configurações de taxa de transmissão definidas de acordo com o protocolo do inversor de 9600 bits por segundo, quantidade de bits por envio, neste caso adotado 7 bits, tipo de paridade neste caso a paridade é par e a quantidade de bits de parada definido como um.

Depois de concluída esta etapa, com o comando SetCommState se configura o estado da comunicação e é feito o teste de que se o resultado desta configuração for um significa que houve erro na configuração, por padrão retorna zero e as funções de validação de erro identificam o ocorrido. Na Figura 3.11 é mostrado o fluxograma da para configuração da transmissão.

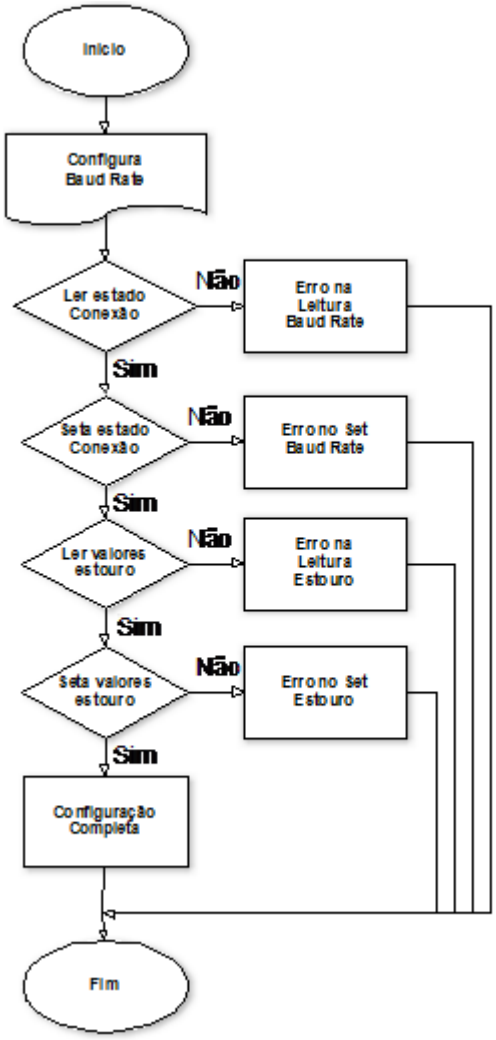


Figura 3.11: Fluxograma para a configuração da transmissão (RS-232).

Outras possibilidades podem ser adotadas e são mostradas as possíveis alterações das configurações de comunicação na tabela 3.1.

Tabela 3.1: Possibilidades de configuração de transmissão.

| BaudRate (Velocidade) | | ByteSize (Tamanho dos dados) | Parity (Paridade) | StopBits (Bits de parada) |
|---|---|---|---|---|
| **CBR_110** | **CBR_19200** | **5** | **EVENPARITY** (par) | **ONESTOPBIT** (1) |
| **CBR_300** | **CBR_38400** | **6** | **MARKPARITY** (marca) | **ONE5STOPBITS**(1.5) |
| **CBR_600** | **CBR_56000** | **7** | **NOPARITY** (nenhuma) | **TWOSTOPBITS** (2) |
| **CBR_1200** | **CBR_57600** | **8** | **ODDPARITY** (ímpar) | |
| **CBR_2400** | **CBR_115200** | | | |
| **CBR_4800** | **CBR_128000** | | | |
| **CBR_9600** | **CBR_256000** | | | |
| **CBR_14400** | | | | |

Por fim, foi criada uma função responsável pelo fechamento da conexão serial é mostrada na Figura 3.12. Se esta função não é executada antes do encerramento do programa, o sistema bloqueia a porta serial e não autoriza mais acessos até que esta função ou o computador onde está o software seja reiniciado.

```
int Fexa_Conexao (HANDLE hCom)
{
    CloseHandle( hCom );  //Fecha a porta aberta anteriormente por CreateFile().
    return 0;
}
```

Figura 3.12: Função responsável pelo encerramento da conexão.

### 3.2.2 Processo de leitura

Para os processos de leitura e escrita foram criadas algumas variáveis auxiliares para o recebimento e envio dos dados, entre elas foi criado a BufferLer que junto com a função WriteFile envia o pedido de leitura de uma variável para o inversor, já a variável BufferRecebe armazena os dados recebidos obtidos via serial usando o comando ReadFile.

Na Figura 3.13 uma rotina de leitura de dados, a função Sleep (15) que gera um delay de 15µs tempo necessário, segundo a WEG, para que seja enviado os dados de resposta pelo inversor. Na Figura 3.14 é mostrado o fluxograma completo dos passos para leitura de um parâmetro.

```
720     BytesEscritos = 0;
        BytesLidos = 0;
        WriteFile( hCom, BufferLer, TamaString, &BytesEscritos, NULL );
        Sleep(15);
        ReadFile( hCom, BufferRecebe, 14, &BytesLidos, NULL );
```

Figura 3.13: Código de telegrama de leitura.

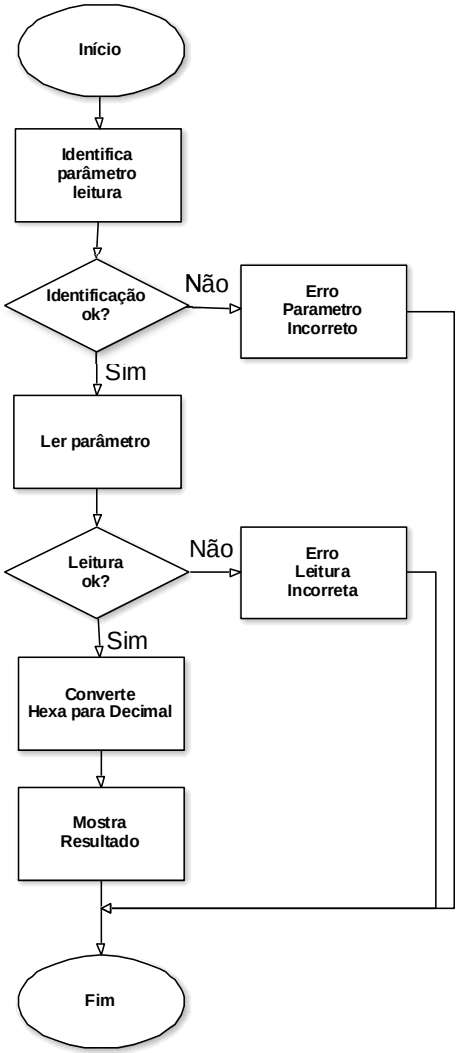


Figura 3.14: Fluxograma de leitura de dados.

### 3.2.3 Processo de escrita

O processo de escrita é semelhante ao de leitura. São utilizadas as mesmas variáveis apenas com diferenciação na quantidade de bits enviados e recebidos, além do tempo de espera entre a escrita e leitura dos dados.

No processo de leitura são enviados 8 bits e são retornados 14 bits, já no processo de escrita são enviado 15 bits e são retornados 2 bits apenas, a Figura 3.15 apresenta a rotina de escrita de dados.

```
790     BytesEscritos = 0;
        BytesLidos = 0;
        WriteFile( hCom, BufferLer, TamaString, &BytesEscritos, NULL );
        Sleep(15);
        ReadFile( hCom, BufferRecebe, 2, &BytesLidos, NULL );
```

Figura 3.15: Código do telegrama de escrita.

Na Figura 3.16 é mostrado o fluxograma com todos os passos utilizados para uma correta escrita de dados.

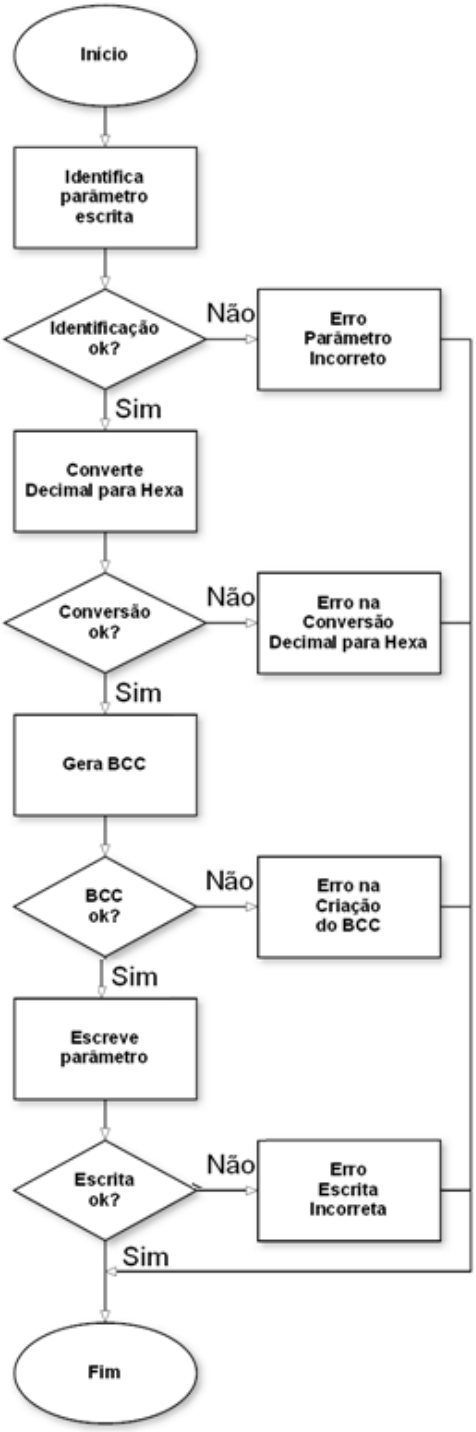


Figura 3.16: Fluxograma de escrita de dados.

## 3.3 FUNÇÕES AUXILIARES

Com o decorrer da programação funções foram criadas para auxiliar no desenvolvimento do sistema, como uma função para conversão de números hexadecimais para decimais, responsável pela conversão do valor recebido pela porta serial para

posterior visualização e entendimento do resultado pelo usuário. Assim como uma função de conversão de decimal para hexadecimal, esta função é responsável pela codificação do valor pretendido pelo usuário para entendimento do inversor. Outra função como o gerador de BCC se fez necessária, gerando o bit de validação do texto a ser enviado.

### 3.3.1 Conversor hexadecimal para decimal

Para obter a leitura dos dados como mostrado na seção 2.6.4.1 os dados recebidos da leitura vem formatados em hexadecimal. Para melhor entendimento do usuário deve-se converter estes dados, para isto foi criada uma função chamada Funcao_HexatoDecimal, na qual como entrada utiliza-se os dados recebidos via serial e no retorno da função tem-se os dados em decimal.

```
//---------------------------------------------------------------------------
int Funcao_HexatoDecimal_Leitura ()
{
    int valor = 0, multiplicador = 0, resultado = 0, cont=8;
    for (cont=8; cont<=11 ; cont++)
    {
        switch (BufferLerResposta[cont])     {
            case (0x00): valor = 0; break;
            case (0x01): valor = 1; break;
            case (0x02): valor = 2; break;
            case (0x03): valor = 3; break;
            case (0x04): valor = 4; break;
            case (0x05): valor = 5; break;
            case (0x06): valor = 6; break;
            case (0x07): valor = 7; break;
            case (0x08): valor = 8; break;
            case (0x09): valor = 9; break;
            case (0x0A): valor = 10; break;
            case (0x0B): valor = 11; break;
            case (0x0C): valor = 12; break;
            case (0x0D): valor = 13; break;
            case (0x0E): valor = 14; break;
            case (0x0F): valor = 15; break;
            default: valor = 5;              }
        switch (cont)                        {
            case (8):  multiplicador = 4096; break;
            case (9):  multiplicador = 256;  break;
            case (10): multiplicador = 16;   break;
            case (11): multiplicador = 1;    break;
            default:                   ;     }
        ShowMessage(IntToHex(valor,1));
    resultado = resultado + (valor * multiplicador);
    }
    return resultado;
}
//---------------------------------------------------------------------------
```

Figura 3.17: Função de conversão de Hexadecimal para decimal.

Com base na Função mostrada na Figura 3.17 se o valor recebido nos quatro bits for respectivamente 0x00, 0x01, 0x06, 0x02 a conversão o valor de 0162 em hexadecimal equivale a 354 na base decimal como mostrado na Tabela 3.2.

Tabela 3.2: Exemplo de conversão de hexadecimal para decimal.

| |
|---|
| resultado = 0<br>valor = 0<br>multiplicador = 4096<br>resultado = 0 + ( 0 * 4096 ) = 0 |
| valor = 1<br>multiplicador = 256<br>resultado = 0 + ( 1 * 256 ) = 256 |
| valor = 6<br>multiplicador = 16<br>resultado = 256 + ( 6 * 16 ) = 352 |
| valor = 2<br>multiplicador = 1<br>resultado = 352 + ( 2 * 1 ) = 354<br>return 354 |

### 3.3.2 Conversor decimal para hexadecimal

Seguindo a idéia de conversão de números em bases diferentes foi gerado um código capaz de converter um valor decimal para quatro dígitos em hexadecimal retornando assim já os valores necessários para envio ao escravo. Na Figura 3.18 apresenta-se como foi gerada a conversão.

```
int Funcao_DecimaltoHexa (int valordecimal)
{
    int quociente01,quociente02,quociente03,quociente04;
    int resto01, resto02, resto03, resto04;

    quociente01 = ( valordecimal / 16 );
    resto01 = (valordecimal%16);

    quociente02 = ( quociente01 / 16 );
    resto02 = (quociente01%16);

    quociente03 = ( quociente02 / 16 );
    resto03 = (quociente02%16);

    //quociente03 . resto03 . resto02 . resto01
```

Figura 3.18: Função de conversão Hexadecimal para Decimal.

As variáveis quociente03, resto03, resto02 e resto01 respectivamente armazenam os valores já convertidos em hexadecimal como mostrado na Tabela 3.3, mostrando a conversão de 354 na base decimal para base hexadecimal resultado em 162 nesta base.

Tabela 3.3: Exemplo de conversão de decimal para hexadecimal.

| |
|---|
| quociente01 = (354 / 16) = 22<br>resto01 = 2 |
| quociente02 = (22 / 16) = 1<br>resto02 = 6<br><br>quociente03 = (1 / 16) = 0<br>resto03 = 1 |
| quociente03*1000+resto03*100+resto02*10+resto01*1<br>0*1000+1*100+6*10+2*1<br>0162 |

### 3.3.3 Gerador BCC

O gerador de Byte de CheCksum é responsável pela validação dos dados enviados para o inversor pelo mestre, esta validação é feita a partir do terceiro até o décimo terceiro bit do pacote de escrita, onde entre estes bits é feito uma lógica chamada *exclusive or* (XOR) esta obtida da eletrônica digita. Na Figura 3.19 é mostrado o diagrama de estados de uma porta lógica XOR [19].

| Entradas | | Saída |
|---|---|---|
| (X) | (Y) | (Z) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Figura 3.19: Diagrama de estados de uma porta XOR.

Na Figura 3.20 verifica-se o código gerado para criação do BCC.

```
int Funcao_GeraBCC ()
{
int aux01, aux02, aux03, aux04, aux05, aux06, aux07, aux08, aux09, aux10, aux11;
int bcc = 0x00;
aux01 = BufferEscrever[3];
aux02 = BufferEscrever[4];
aux03 = BufferEscrever[5];
aux04 = BufferEscrever[6];
aux05 = BufferEscrever[7];
aux06 = BufferEscrever[8];
aux07 = BufferEscrever[9];
aux08 = BufferEscrever[10];
aux09 = BufferEscrever[11];
aux10 = BufferEscrever[12];
aux11 = BufferEscrever[13];

bcc=aux01^aux02^aux03^aux04^aux05^aux06^aux07^aux08^aux09^aux10^aux11;

ShowMessage(aux01);
ShowMessage(aux02);
ShowMessage(bcc);
BufferEscrever[14] = bcc;

return 0;
}
```

Figura 3.20: Função de geração do BCC.

A linguagem se mostrou extremamente útil nesta etapa do desenvolvimento, uma vez que o símbolo do circunflexo (^) entre variáveis do tipo inteiro representa a XOR na programação, assim foi feito esta comparação entre todos os valores do vetor de escrita e o retorno foi o valor do BCC [19][20].

# 4 RESULTADOS E ANÁLISES

## 4.1 INTERFACES

Resolvidas as dificuldades geradas pela comunicação e configuração dos parâmetros necessários para leitura e escrita, começou-se a trabalhar a interface que apareceria ao usuário. Uma interface bem acessível torna o trabalho e a recuperação de dados mais eficiente melhorando assim o processo de produção.

Foi proposto ao trabalho gerar duas telas de monitoramento, a primeira mostrada na Figura 4.1 mostra a primeira tela. Esta criada para mostrar o que poderia ser executado no sistema, como a configuração da comunicação, leitura e escrita de parâmetros além de controle de velocidade e sentido de giro do motor.

Figura 4.1: Primeira tela para monitoramento.

Na segunda tela são mostrados gráficos de desempenhos de cada parâmetro escolhido pelo usuário, assim este poderá comparar visualmente o desempenho de cada ponto escolhido por ele. Esta tela está representada na Figura 4.2

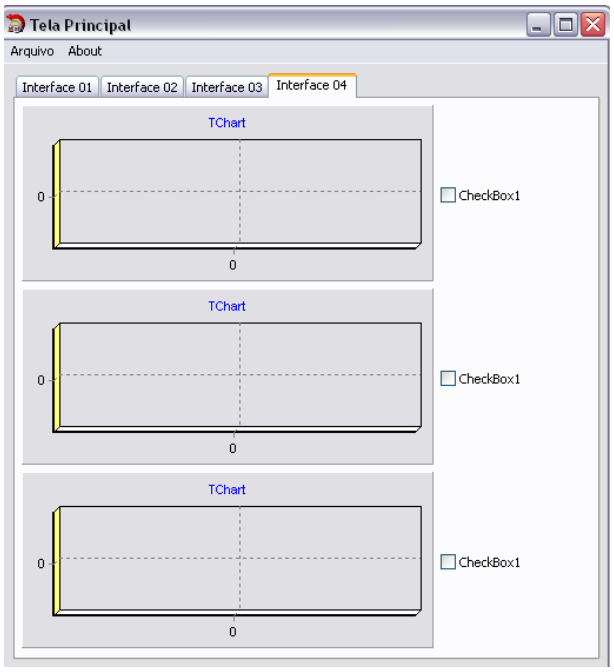


Figura 4.2: Segunda tela de monitoramento.

## 4.2 TESTES

Para testes de desempenho foi utilizado o software *Device Monitoring Studio*, este software analisou o que foi enviado e recebido pela porta serial enquanto foi enviado pedidos de leitura e escrita para o inversor. Na Figura 4.3 é mostrada a tela do software monitorando a leitura de um parâmetro.

Na Figura 4.4 o detalhamento da escrita do parâmetro na porta serial e o retorno obtido, neste caso não ouve resposta do inversor, com isto pode-se verificar que o dado esta sendo enviado corretamente como mostrado o código “□01801□” descrevendo que esta tentando ser lido o parâmetro P104 do inversor os □ são parâmetros que não são identificados pelo software como os caracteres de controle.

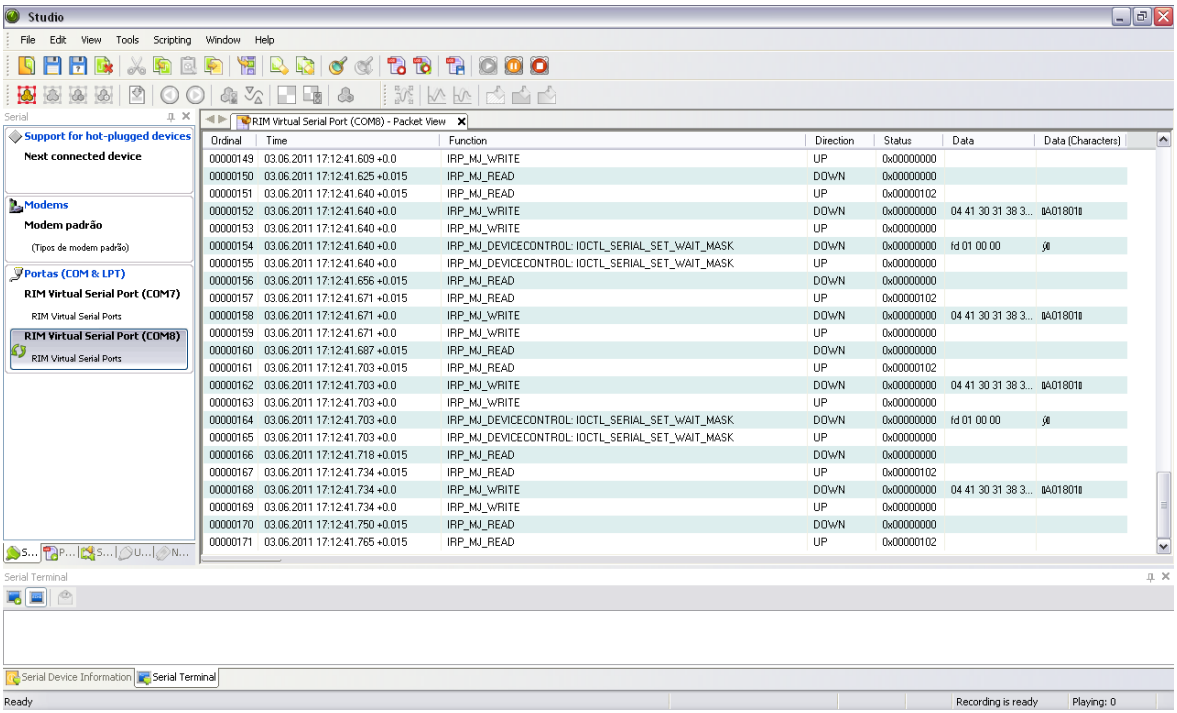


Figura 4.3: Software *Device Monitoring Studio* monitorando o tráfego na porta serial.

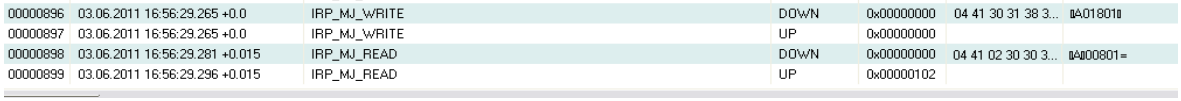


Figura 4.4: Detalhamento do monitoramento de escrita e leitura de dados.

Resolvido os problemas relacionados ao tempo de leitura e escrita na Figura 4.5 é mostrada a leitura de um parâmetro e a resposta obtida pela serial, pode-se notar "□A□00801=" indicando o valor do parâmetro lido no caso P001 e o valor deste parâmetro mostrado depois do "=" este dado pode ser lido no buffer de armazenamento da resposta de leitura chamado BufferLerResposta.

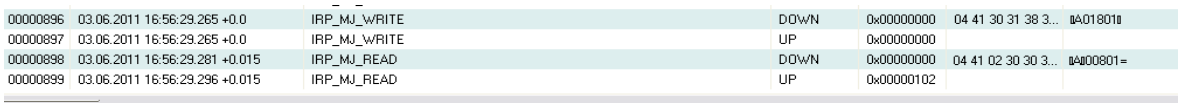


Figura 4.5: Exemplo de leitura e resposta via comunicação serial.

Para validação da função de conversão de hexadecimal para decimal foi analisado a resposta mostrado na interface do usuário como mostrado na Figura 4.6 o valor escrito na tela está corretamente convertido para decimal. Neste caso o valor lido foi convertido para

decimal e obteve-se o valor de 333, neste caso representa 333 rotações por minuto já que o parâmetro P001 é relativo a velocidade do motor.

Figura 4.6: Retorno da leitura lida convertido em decimal mostrado no campo com índice 333.

Os mesmos testes para escrita de dados foram executados e obteve-se os mesmos resultados mostrando também a execução das rotinas de conversão de decimal para hexadecimal e geração do BCC.

## 4.3 VALIDAÇÃO DO SISTEMA

Para a validação do sistema, foram testados vários comandos possíveis no sistema. Na Figura 4.7 é mostrada em “funções básicas”, a velocidade em que esta o motor neste caso 965 rotações por minuto, e a leitura feita do parâmetro P002 obteve o mesmo resultado, ou seja, 965 mostrando que a leitura do parâmetro e a conversão de hexadecimal para decimal estão ocorrendo uma vez que os dados lidos são sempre em hexadecimal.

Ainda analisando a Figura 4,7 pode-se observar que o processo de escrita também ocorre de forma correta uma vez que os parâmetros das “Funções Básicas” são todos de escrita no inversor, validando assim o sistema proposto.

Figura 4.7: Interface principal com o software funcionando.

Na Figura 4.8 é mostrada outra possibilidade de acesso às informações, os gráficos gerados poderiam ser de tensão, corrente e velocidade, foram escolhidas: corrente e velocidade. O software desenvolvido possui recursos que facilita a geração de gráficos, no protótipo desenvolvido podendo-se obter gráficos referente a tensão, corrente e velocidade do motor.

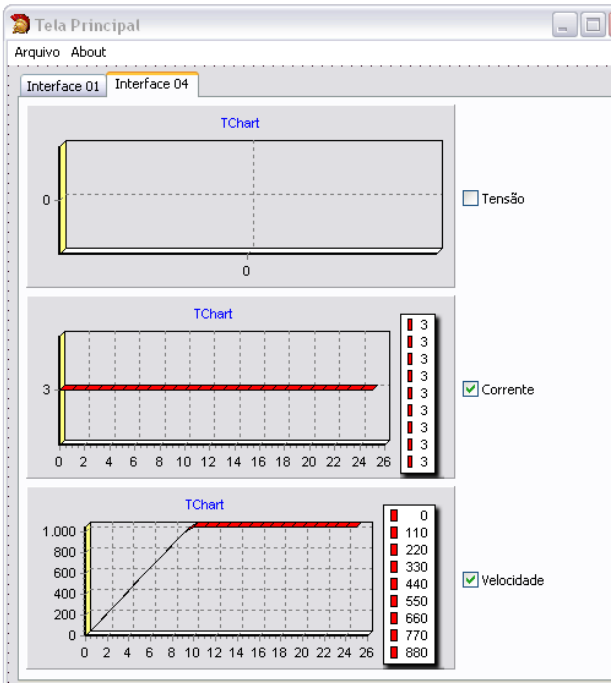


Figura 4.8. Tela de gráficos com software funcionando.

## 4.4 DOCUMENTAÇÃO

No Apêndice A foi criado um pequeno manual com as instruções básicas de uso do software proposto, este manual possui referências sobre os requisitos mínimos de uma máquina para rodar o programa, as configurações necessárias antes de se usar e uma breve descrição de cada parte do software.

No Apêndice B é descrito do código fonte gerado pelo software, ele é composto de todas as rotinas e sub-rotinas criadas para execução perfeita execução do programa desenvolvido.

# 5 CONCLUSÕES

## 5.1 CONSIDERAÇÕES FINAIS

O presente trabalho alcançou os objetivos propostos e mostrou que há a possibilidade de criação de um sistema diferenciado, adaptado as necessidades de cada projeto, assim não restringindo o usuário a utilização apenas do software desenvolvido pelo fabricante do equipamento.

Os protocolos de comunicação utilizados se mostraram eficientes para a comunicação com baixa taxa de erro entre os equipamentos, contudo não permitem a criação de conexões simultâneas, neste caso não é permitida a utilização do software desenvolvido ao mesmo tempo em que se utiliza o programa fornecido pelo fabricante.

As rotinas desenvolvidas podem ser utilizadas para comunicar com outros dispositivos desde que estes utilizem as mesmas configurações de comunicação descritas na tabela 4.1 e o protocolo de comunicação baseado no tipo pergunta/resposta conforme a ISO 1745 e ISO 646.

O software desenvolvido pode ser utilizado para fins didáticos devido ao seu fácil manuseio, auxiliando no aprendizado em técnicas de monitoramento e controle de sistemas tanto com sua interface quanto ao seu código fonte disponibilizado gratuitamente.

Este trabalho proporciona uma série de possibilidades para continuação e geração de novos resultados, estas são descritas no item 5.1, já no item 5.2 é descrito as publicações feitas com base no trabalho proposto.

## 5.2 TRABALHOS FUTUROS

O software foi desenvolvido utilizando a porta de comunicação RS-232 e como continuidade do trabalho pode-se citar como possibilidades:

- Agregar novas funções como técnicas de controle via *fuzzy* ou PID;
- Transcrever o código para outras linguagens acessíveis em outras plataformas de sistemas operacionais;
- Utilizar o código para criação de sistemas para monitoramento remoto;
- Baseado no processo de leitura e escrita criar novos protocolos de comunicação proprietários para novos projetos.

As possibilidades de uso do software são diversas, ficando a cargo da capacidade do programador saber onde, quando e como utilizá-las.

## 5.3 PUBLICAÇÃO

O presente trabalho foi submetido e aceito para participação do congresso do ICECE 2011 – VII Conferência Internacional de Educação em Engenharia e Computação que será realizado em Guimarães, localizada em Portugal, entre os dias 25 e 28 de setembro de 2011.

Neste ano o tema deste congresso é “Educação em Engenharia Inspirando a Próxima Geração de Engenheiros” tema muito propício uma vez que um dos focos deste projeto foi a criação de um software que pudesse ser utilizado no meio acadêmico.

No Apêndice C contém uma cópia do artigo, este que foi aceito pela comissão organizadora do evento ele segue exatamente os padrões de formatação pedidos pela comissão técnica do congresso.

## REFERÊNCIAS BIBLIOGRÁFICAS


[1] WEI, S. *et al*. Design And Implementation Of Small-Scale Industrial Computer Monitoring System. In: 2nd INTERNATIONAL CONFERENCE ON MECHANICAL AND ELECTRONICS ENGINEERING (1-3 Agu, 2010, Tokyo, Japão). p. 189-192

[2] Santos, E.A.P.. *et al*. Proposal Of An Industrial Information System Model For Automatic. In: 2nd IEEE INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (15-18 Sep, 2008,Hamburg, Alemanha). p. 436-439

[3] TANENBAUM, A.S. **Sistemas operacionais modernos**, 2 ed. São Paulo: Pearson Prentice Hall, 2003.

[4] SILVA, M.G. **Informática : Terminologia Básica : Microsoft Windows XP, Word XP, Excel XP, Access XP, PowerPoint XP**, 4 ed. São Paulo: Érica, 2008.

[5] HUNT, C. **Servidores de Redes com Linux: O recurso essencial para administradores de sistemas**. 2 ed. São Paulo: Market Books 2000.

[6] BALL, B. *et. al*. **Dominando Red Hat Linux 7**. 1 ed. São Paulo: Ciência Moderna, 2002.

[7] POGUE, D. **Mac OS X Leopard**, 1 ed. São Paulo: Universo dos Livros, 2008.

[8] Denver. A. "*Serial Communications in Win32*", Disponível em: <http://msdn.microsoft.com/en-us/library/ms810467#serial_topic4>. Acesso em: 14/01/2011.

[9] Elipse Scada – "Manual do usuário". Disponível em: < http://www.elipse.com.br/get/Elipse%20SCADA%5Cv2.29%5CBuild095%5Cscadamanual_br.pdf > . Acesso em: 01/02/2011.

[10] RsLinx – "Training Guide". Disponível em: < http://www.rockwellsoftware.com/download/training/detinfo/rst-lnx-tgde.pdf >. Acesso em 01/02/2011.

[11] WinCC – "Configuration Manual". Disponível em: < http://cache.automation.siemens.com/dnl/jg/jgxMTE3AAAA_1145270_HB/Conf m1_e.pdf >. Acesso em 01/02/2011.

[12] DEITEL, L, **C++ Como Programar.** 5 ed. São Paulo: Prentice Hall, 2006.

[13] STROUSTRUP, B. ***The Design and Evolution of C++.*** Addison-Wesley, 1994.

[14] WEG, “**Manual do inversor de freqüência CFW – 09**”, Disponível em: <http://www.weg.net/files/products/WEG-cfw-09-manual-do-usuario-0899.5298-4.0x-manual-portugues-br.pdf> . Acesso em: 05/04/2010.

[15] CodeProject. Disponível em: <http://www.codeproject.com/>. Acesso em: 10 de dezembro de 2011.

[16] Eterlogic.com. Disponível em: <http://www.eterlogic.com/>. Acesso em: 08 de janeiro de 2011.

[17] CommFront Communications. Disponível em: <http://www.232analyzer.com/232default.htm>. Acesso em: 08 de janeiro de 2011.

[18] HDD Software. Disponível em: <http://www.hhdsoftware.com/>. Acesso em: 08 de janeiro de 2011.

[19] IDOETA, I. V.; CAPUANO, F. G.; **Elementos de eletrônica Digital**. 40 ed. São Paulo: Érica. 2007.

[20] ALVES, T. L. **Curso Básico de C++ Builder**. Disponível em: <http://www.dicasbcb.com.br/>. Acesso em: 20 de novembro de 2010.

# APÊNDICE A – Manual do usuário

**Supervisor I.F.**

**Manual do Usuário**

ÍNDICE


| | | |
|---|---|---|
| 1. | Introdução | 1 |
| 2. | Pré-Requisitos | 1 |
| 2.1 | Sistema Operacional | 2 |
| 3. | Instalação | 2 |
| 4. | Interface | 3 |
| 5. | Código Fonte | 7 |


## 1. Introdução

Leia o manual de instruções, este foi elaborado para facilitar sua vida. Antes de usar verifique se seu equipamento está conectado corretamente e se seu computador atende os requisitos mínimos para uso.

## 2. Pré-Requisitos

Estes são os requisitos que uma máquina deve satisfazer para que o Supervisor I.F. tenha um desempenho satisfatório.

Quanto mais poderosa for a máquina melhor será o desempenho do sistema.

| | Mínimo | Recomendado |
|---|---|---|
| Processador | Pentium III 800MHz | Pentium IV 1.4GHz |
| Memória | 128 Mb | 512 Mb |
| Portas de comunicação | Serial | Serial |

2.1 Sistema Operacional

O Supervisor I.F. funciona nos seguintes sistemas operacionais: Windows 98, Windows 2000, Windows XP, Windows 2003, Windows Vista e Windows 7.

3. Instalação

O Supversiro I.F. vem configurado e pronto para ser executável é apenas necessário a execução do arquivo SupervisorIF.exe

4. Interface

A primeira tela quando se inicia o programa é descrita na Figura A1 nela é mostrada os principais recursos disponíveis pelo software.

Figura A1: Tela principal do Supervisor I.F.

Na parte de relacionada à COMUNICAÇÃO existe a opção de conectar, desconectar e definir qual porta de comunicação será utilizado, verifique seu dispositivo em qual porta esta conectada antes de decidir conectar o sistema.

Na figura A2 são mostrados os campos relativos á LEITURA e ESCRITA de parâmetros, quando se faz a leitura de um parâmetro o resultado é mostrado abaixo do botão LER, e quando é feito a escrita de um parâmetro o resultado é mostrado em uma tela popup com a confirmação ou não da escrita.

Figura A2: Campos para leitura e escrita de dados.

Finalmente na figura A3 tem-se as FUNÇÕES BÁSICAS de acesso, neste caso de ligar, desligar o motor assim como sentidos da rotação sendo horário ou anti-horário e uma barra de seleção de velocidade onde pode-se variar de ponto a ponto a velocidade do motor e ao seu lado o botar de SETAR a velocidade escolhida.

Figura A3: Funções básicas de acesso.

A segunda aba chamada INTERFACE 02 é mostrada na Figura A4. Nesta interface há à possibilidade de selecionar entre três parâmetros são eles: Tensão, Corrente e Velocidade do sistema monitorado.

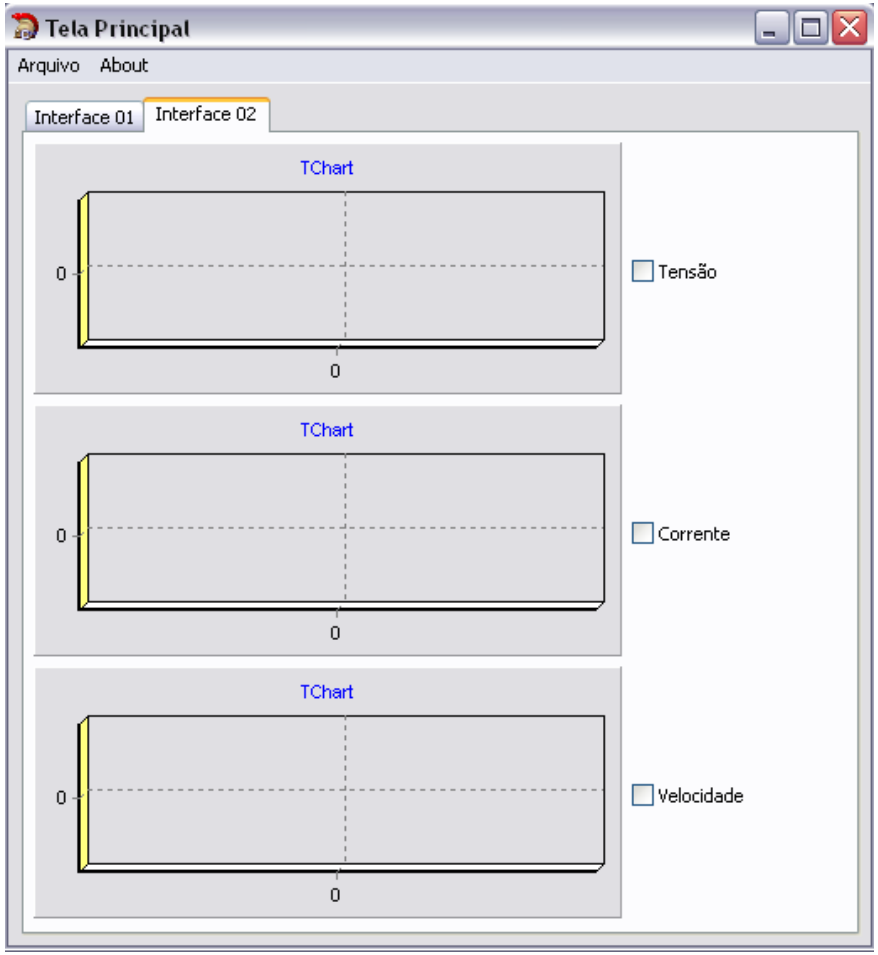


Figura A4: Tela secundaria do Supervisor I.F.

## 5. Código Fonte

O código fonte feito em C/C++ está disponível na pasta CÓDIGO FONTE junto com o programa nele existem todas as funções e bibliotecas criadas e estão disponíveis para interpretação, codificação e/ou criação de novos software.

# APÊNDICE B - Código Fonte

**Arquivo Funcoes.cpp**

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <assert.h>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <fstream.h>
#include <stdlib.h>
#include <winNT.h>
#include <windows.h.>
#include <winbase.h>
#include <dos.h>
#include "Form_interface.h"
#include "Funcoes.h"
//---------------------------------------------------------------------------
HANDLE  hCom; // Handle para a Porta Serial (identificador).
DCB dcb; //Estrutura utilizada para definir todos os parâmetros da comunicação.
COMMTIMEOUTS CommTimeouts;
DWORD BytesLidos = 0;
DWORD BytesEscritos = 0;
//DWORD dwEvtMask;
int TamaString;    //Para armazenar o tamanho da String.

char BufferLer[10];   //Para armazenar a string a ser enviada.
char BufferLerResposta[16];
char BufferEscrever[17];
char BufferEscreverResposta[2];

//---------------------------------------------------------------------------
int Cria_Conexao (char PortaComunica)
{
        char Porta[7];
        switch (PortaComunica)
        {
                case 1: strcpy (Porta,"COM1"); break;
                case 2: strcpy (Porta,"COM2"); break;
                case 3: strcpy (Porta,"COM3"); break;
                case 4: strcpy (Porta,"COM4"); break;
                case 5: strcpy (Porta,"COM5"); break;
                case 6: strcpy (Porta,"COM6"); break;
                case 7: strcpy (Porta,"COM7"); break;
                case 8: strcpy (Porta,"COM8"); break;
                case 9: strcpy (Porta,"COM9"); break;
                default:strcpy (Porta,"COM1"); break;
        }
        char *NomePorta = Porta;
        hCom = CreateFile
        (
                NomePorta,   //Nome da porta.
                GENERIC_READ | GENERIC_WRITE,   //Para leitura e escrita.
                0,       //Nenhuma outra abertura será permitida.
                NULL,  //Atributos de segurança. (NULL) padrão.
                OPEN_EXISTING,  //Criação ou abertura.
                0, //Entrada e saída sem overlapped.
                NULL     //Atributos e Flags. Deve ser NULL para COM.
        );
        if(hCom == INVALID_HANDLE_VALUE)
        {
                return 1;   //Erro ao tentar abrir a porta especificada.
        }
        return 0;
}
//---------------------------------------------------------------------------
```

```
int Cria_BaudRate ()
{
	if( GetCommState(hCom, &dcb) == 0 )
	{
		return 1;  //// Erro na leitura de DCB.
	}
	dcb.BaudRate = CBR_9600;
	dcb.ByteSize = 7;
	dcb.Parity = EVENPARITY;
	dcb.StopBits = ONESTOPBIT;
	//Define novo estado.
	if( SetCommState(hCom, &dcb) == 0 )
	{
		return 1;   //Erro na escrita de DCB.
	}
	if (GetCommTimeouts(hCom, &CommTimeouts) == 0)
	{
		ShowMessage("Erro: GetCommTimeouts" );
	}
	CommTimeouts.ReadIntervalTimeout                = 30       ;
	CommTimeouts.ReadTotalTimeoutMultiplier         = 0                ;
	CommTimeouts.ReadTotalTimeoutConstant           = 15       ;
	CommTimeouts.WriteTotalTimeoutMultiplier        = 0                ;
	CommTimeouts.WriteTotalTimeoutConstant          = 3                ;

	if (SetCommTimeouts(hCom, &CommTimeouts) == 0)
	{
		ShowMessage("Erro: SetCommTimeouts" );
	}
	return 0;
}
//---------------------------------------------------------------------------
int Fexa_Conexao ()
{
	CloseHandle( hCom ); //Fecha a porta aberta anteriormente por CreateFile().
	return 0;
}
//---------------------------------------------------------------------------
int Funcao_Leitura ()
{
	int contador=0;
	do
	{
			contador++;
		BytesEscritos = 0;
		BytesLidos = 0;
		TamaString = strlen(BufferLer);
		WriteFile( hCom, BufferLer, TamaString, &BytesEscritos, NULL );
			SetCommMask( hCom, 0x01fd);
			Sleep(15);
		ReadFile( hCom, BufferLerResposta, 14, &BytesLidos, NULL );
//---------------------------------------------------------------------------

		if( (BufferLer[2]!=BufferLerResposta[2])||
			(BufferLer[5]!=BufferLerResposta[5])||
			(BufferLer[6]!=BufferLerResposta[6])||
			(BufferLer[7]!=BufferLerResposta[7]) )
		 {

		 }
		if( (BufferLer[2]==BufferLerResposta[2])&&
			(BufferLer[5]==BufferLerResposta[5])&&
			(BufferLer[6]==BufferLerResposta[6])&&
			(BufferLer[7]==BufferLerResposta[7])&&
```

```
                                (contador>=1) )
                        {
                        return 0;
                        }
        } while   ( (contador<10)
                                );
        return 1;
}
//---------------------------------------------------------------------------
int Funcao_Escrita (int val, int param)
{
                BufferEscrever[0] = 0x04;
                BufferEscrever[1] = 0x41;
                BufferEscrever[2] = 0x02;
                BufferEscrever[5] = 0x30;
                BufferEscrever[6] = 0x30;
                BufferEscrever[7] = 0x38;
                BufferEscrever[8] = 0x30;
                BufferEscrever[9] = 0x34;
                BufferEscrever[10] = 0x3d;
        if ((val == 1)&&(param == 1)) //motor liga
        {
                BufferEscrever[0] = 0x04;
                BufferEscrever[1] = 0x41;
                BufferEscrever[2] = 0x02;
                BufferEscrever[5] = 0x30;
                BufferEscrever[6] = 0x30;
                BufferEscrever[7] = 0x38;
                BufferEscrever[8] = 0x30;
                BufferEscrever[9] = 0x33;
                BufferEscrever[10] = 0x3d;
                BufferEscrever[11] = 0x00;
                BufferEscrever[12] = 0x03;
                BufferEscrever[13] = 0x00;
                BufferEscrever[14] = 0x03;
                BufferEscrever[15] = 0x03;
                BufferEscrever[16] = 0x05;
        } else if ((val == 0)&&(param == 1))//motor desliga
        {
                BufferEscrever[0] = 0x04;
                BufferEscrever[1] = 0x41;
                BufferEscrever[2] = 0x02;
                BufferEscrever[5] = 0x30;
                BufferEscrever[6] = 0x30;
                BufferEscrever[7] = 0x38;
                BufferEscrever[8] = 0x30;
                BufferEscrever[9] = 0x33;
                BufferEscrever[10] = 0x3D;
                BufferEscrever[11] = 0x00;
                BufferEscrever[12] = 0x01;
                BufferEscrever[13] = 0x00;
                BufferEscrever[14] = 0x00;
                BufferEscrever[15] = 0x03;
                BufferEscrever[16] = 0x04;
        } else if ((val == 1)&&(param == 0)) //motor horario
        {
                BufferEscrever[0] = 0x04;
                BufferEscrever[1] = 0x41;
                BufferEscrever[2] = 0x02;
                BufferEscrever[5] = 0x30;
                BufferEscrever[6] = 0x30;
                BufferEscrever[7] = 0x38;
                BufferEscrever[8] = 0x30;
                BufferEscrever[9] = 0x33;
```

```
                BufferEscrever[10] = 0x3d;
                BufferEscrever[11] = 0x00;
                BufferEscrever[12] = 0x04;
                BufferEscrever[13] = 0x00;
                BufferEscrever[14] = 0x04;
                BufferEscrever[15] = 0x03;
                BufferEscrever[16] = 0x05;
        } else if ((val == 0)&&(param == 0)) //motor anti horario
        {
                BufferEscrever[0] = 0x04;
                BufferEscrever[1] = 0x41;
                BufferEscrever[2] = 0x02;
                BufferEscrever[5] = 0x30;
                BufferEscrever[6] = 0x30;
                BufferEscrever[7] = 0x38;
                BufferEscrever[8] = 0x30;
                BufferEscrever[9] = 0x33;
                BufferEscrever[10] = 0x3D;
                BufferEscrever[11] = 0x00;
                BufferEscrever[12] = 0x04;
                BufferEscrever[13] = 0x00;
                BufferEscrever[14] = 0x00;
                BufferEscrever[15] = 0x03;
                BufferEscrever[16] = 0x01;
        }
        int contador=0, acerto=0, erro=0;
        do
        {
                contador++;
                BytesEscritos = 0;
                BytesLidos = 0;
                TamaString = strlen(BufferEscrever);
                TamaString = 15;
                WriteFile( hCom, BufferEscrever, TamaString, &BytesEscritos, NULL );
                Sleep(15);
                ReadFile( hCom, BufferEscreverResposta, 2, &BytesLidos, NULL );
                if (BufferEscreverResposta[1]==0x06)
                {
                        acerto++;
                } else if (BufferEscreverResposta[1]==0x15)
                {
                        erro++;
                }
                if (acerto == 3)
                {
                        return 0;
                } else if (erro == 3)
                {
                        return 1;
                }
        } while (contador<10);

        return 1;
}
//---------------------------------------------------------------------------
int Funcao_HexatoDecimal_Leitura ()
{
        int valor = 0, multiplicador = 0, resultado = 0, cont=8;
        for (cont=8; cont<=11 ; cont++)
        {
                switch (BufferLerResposta[cont])
                {
                        case (0x00): valor = 0; break;
                        case (0x01): valor = 1; break;
```

```
                        case (0x02): valor = 2; break;
                        case (0x03): valor = 3; break;
                        case (0x04): valor = 4; break;
                        case (0x05): valor = 5; break;
                        case (0x06): valor = 6; break;
                        case (0x07): valor = 7; break;
                        case (0x08): valor = 8; break;
                        case (0x09): valor = 9; break;
                        case (0x0A): valor = 10; break;
                        case (0x0B): valor = 11; break;
                        case (0x0C): valor = 12; break;
                        case (0x0D): valor = 13; break;
                        case (0x0E): valor = 14; break;
                        case (0x0F): valor = 15; break;
                        default: valor = 5;
                }
                switch (cont)
                {
                        case (8):  multiplicador = 4096; break;
                        case (9):  multiplicador = 256;  break;
                        case (10): multiplicador = 16;   break;
                        case (11): multiplicador = 1;    break;
                        default:                 ;
                }
                ShowMessage(IntToHex(valor,1));
        resultado = resultado + (valor * multiplicador);
        }
        return resultado;
}
//---------------------------------------------------------------------------
int Funcao_DecimaltoHexa (int valordecimal)
{
        int quociente01,quociente02,quociente03,quociente04;
        int resto01, resto02, resto03, resto04;
        quociente01 = ( valordecimal / 16 );
        resto01 = (valordecimal%16);
        quociente02 = ( quociente01 / 16 );
        resto02 = (quociente01%16);
        quociente03 = ( quociente02 / 16 );
        resto03 = (quociente02%16);
        //quociente03 . resto03 . resto02 . resto01
        if (quociente03 == 0)  { BufferEscrever[11] = 0x00; }
        if (quociente03 == 1)  { BufferEscrever[11] = 0x01; }
        if (quociente03 == 2)  { BufferEscrever[11] = 0x02; }
        if (quociente03 == 3)  { BufferEscrever[11] = 0x03; }
        if (quociente03 == 4)  { BufferEscrever[11] = 0x04; }
        if (quociente03 == 5)  { BufferEscrever[11] = 0x05; }
        if (quociente03 == 6)  { BufferEscrever[11] = 0x06; }
        if (quociente03 == 7)  { BufferEscrever[11] = 0x07; }
        if (quociente03 == 8)  { BufferEscrever[11] = 0x08; }
        if (quociente03 == 9)  { BufferEscrever[11] = 0x09; }
        if (quociente03 == 10) { BufferEscrever[11] = 0x0A; }
        if (quociente03 == 11) { BufferEscrever[11] = 0x0B; }
        if (quociente03 == 12) { BufferEscrever[11] = 0x0C; }
        if (quociente03 == 13) { BufferEscrever[11] = 0x0D; }
        if (quociente03 == 14) { BufferEscrever[11] = 0x0E; }
        if (quociente03 == 15) { BufferEscrever[11] = 0x0F; }
        if (resto03 == 0)  { BufferEscrever[12] = 0x00; }
        if (resto03 == 1)  { BufferEscrever[12] = 0x01; }
        if (resto03 == 2)  { BufferEscrever[12] = 0x02; }
        if (resto03 == 3)  { BufferEscrever[12] = 0x03; }
        if (resto03 == 4)  { BufferEscrever[12] = 0x04; }
        if (resto03 == 5)  { BufferEscrever[12] = 0x05; }
        if (resto03 == 6)  { BufferEscrever[12] = 0x06; }
```

```
            if (resto03 == 7)  { BufferEscrever[12] = 0x07; }
            if (resto03 == 8)  { BufferEscrever[12] = 0x08; }
            if (resto03 == 9)  { BufferEscrever[12] = 0x09; }
            if (resto03 == 10) { BufferEscrever[12] = 0x0A; }
            if (resto03 == 11) { BufferEscrever[12] = 0x0B; }
            if (resto03 == 12) { BufferEscrever[12] = 0x0C; }
            if (resto03 == 13) { BufferEscrever[12] = 0x0D; }
            if (resto03 == 14) { BufferEscrever[12] = 0x0E; }
            if (resto03 == 15) { BufferEscrever[12] = 0x0F; }
            if (resto02 == 0)  { BufferEscrever[13] = 0x00; }
            if (resto02 == 1)  { BufferEscrever[13] = 0x01; }
            if (resto02 == 2)  { BufferEscrever[13] = 0x02; }
            if (resto02 == 3)  { BufferEscrever[13] = 0x03; }
            if (resto02 == 4)  { BufferEscrever[13] = 0x04; }
            if (resto02 == 5)  { BufferEscrever[13] = 0x05; }
            if (resto02 == 6)  { BufferEscrever[13] = 0x06; }
            if (resto02 == 7)  { BufferEscrever[13] = 0x07; }
            if (resto02 == 8)  { BufferEscrever[13] = 0x08; }
            if (resto02 == 9)  { BufferEscrever[13] = 0x09; }
            if (resto02 == 10) { BufferEscrever[13] = 0x0A; }
            if (resto02 == 11) { BufferEscrever[13] = 0x0B; }
            if (resto02 == 12) { BufferEscrever[13] = 0x0C; }
            if (resto02 == 13) { BufferEscrever[13] = 0x0D; }
            if (resto02 == 14) { BufferEscrever[13] = 0x0E; }
            if (resto02 == 15) { BufferEscrever[13] = 0x0F; }
            if (resto01 == 0)  { BufferEscrever[14] = 0x00; }
            if (resto01 == 1)  { BufferEscrever[14] = 0x01; }
            if (resto01 == 2)  { BufferEscrever[14] = 0x02; }
            if (resto01 == 3)  { BufferEscrever[14] = 0x03; }
            if (resto01 == 4)  { BufferEscrever[14] = 0x04; }
            if (resto01 == 5)  { BufferEscrever[14] = 0x05; }
            if (resto01 == 6)  { BufferEscrever[14] = 0x06; }
            if (resto01 == 7)  { BufferEscrever[14] = 0x07; }
            if (resto01 == 8)  { BufferEscrever[14] = 0x08; }
            if (resto01 == 9)  { BufferEscrever[14] = 0x09; }
            if (resto01 == 10) { BufferEscrever[14] = 0x0A; }
            if (resto01 == 11) { BufferEscrever[14] = 0x0B; }
            if (resto01 == 12) { BufferEscrever[14] = 0x0C; }
            if (resto01 == 13) { BufferEscrever[14] = 0x0D; }
            if (resto01 == 14) { BufferEscrever[14] = 0x0E; }
            if (resto01 == 15) { BufferEscrever[14] = 0x0F; }
            return 0;
}
//---------------------------------------------------------------------------
int Funcao_GeraBCC ()
{
int aux01, aux02, aux03, aux04, aux05, aux06, aux07, aux08, aux09, aux10, aux11;
int bcc = 0x00;
aux01 = BufferEscrever[5];
aux02 = BufferEscrever[6];
aux03 = BufferEscrever[7];
aux04 = BufferEscrever[8];
aux05 = BufferEscrever[9];
aux06 = BufferEscrever[10];
aux07 = BufferEscrever[11];
aux08 = BufferEscrever[12];
aux09 = BufferEscrever[13];
aux10 = BufferEscrever[14];
aux11 = BufferEscrever[15];
bcc=aux01^aux02^aux03^aux04^aux05^aux06^aux07^aux08^aux09^aux10^aux11;
ShowMessage(aux01);
ShowMessage(aux02);
ShowMessage(bcc);
BufferEscrever[16] = bcc;
```

```
return 0;
}
//---------------------------------------------------------------------------
int Funcao_Identifica_Parametro_Leitura (int parametro)
{
	int aux=0;
	strcpy(BufferLer, "\x04""A""01800""\x05");
	if ((parametro<100))		{	BufferLer[5] = '1'; aux =   0; }
	if ((parametro>=100) && (parametro<=199)) { BufferLer[5] = '2'; aux = 100; }
	if ((parametro>=200) && (parametro<=299)) { BufferLer[5] = '3'; aux = 200; }
	if ((parametro>=300) && (parametro<=399)) { BufferLer[5] = '4'; aux = 300; }
	if ((parametro>=400) && (parametro<=499)) { BufferLer[5] = '5'; aux = 400; }
	if ((parametro>=500) && (parametro<=599)) { BufferLer[5] = '6'; aux = 500; }
	if ((parametro>=600) && (parametro<=699)) { BufferLer[5] = '7'; aux = 600; }

	if(aux == 0)
	{
		if( ( parametro/10 ) == 0 ) { BufferLer[7]= '0'; }
		if( ( parametro/10 ) == 1 ) { BufferLer[7]= '1'; }
		if( ( parametro/10 ) == 2 ) { BufferLer[7]= '2'; }
		if( ( parametro/10 ) == 3 ) { BufferLer[7]= '3'; }
		if( ( parametro/10 ) == 4 ) { BufferLer[7]= '4'; }
		if( ( parametro/10 ) == 5 ) { BufferLer[7]= '5'; }
		if( ( parametro/10 ) == 6 ) { BufferLer[7]= '6'; }
		if( ( parametro/10 ) == 7 ) { BufferLer[7]= '7'; }
		if( ( parametro/10 ) == 8 ) { BufferLer[7]= '8'; }
		if( ( parametro/10 ) == 9 ) { BufferLer[7]= '9'; }
		if( ( parametro%10 ) == 0 ) { BufferLer[8]= '0'; }
		if( ( parametro%10 ) == 1 ) { BufferLer[8]= '1'; }
		if( ( parametro%10 ) == 2 ) { BufferLer[8]= '2'; }
		if( ( parametro%10 ) == 3 ) { BufferLer[8]= '3'; }
		if( ( parametro%10 ) == 4 ) { BufferLer[8]= '4'; }
		if( ( parametro%10 ) == 5 ) { BufferLer[8]= '5'; }
		if( ( parametro%10 ) == 6 ) { BufferLer[8]= '6'; }
		if( ( parametro%10 ) == 7 ) { BufferLer[8]= '7'; }
		if( ( parametro%10 ) == 8 ) { BufferLer[8]= '8'; }
		if( ( parametro%10 ) == 9 ) { BufferLer[8]= '9'; }
	}
	else
	{
		if( ( aux-parametro/10 ) == 0 ) { BufferLer[7]= '0'; }
		if( ( aux-parametro/10 ) == 1 ) { BufferLer[7]= '1'; }
		if( ( aux-parametro/10 ) == 2 ) { BufferLer[7]= '2'; }
		if( ( aux-parametro/10 ) == 3 ) { BufferLer[7]= '3'; }
		if( ( aux-parametro/10 ) == 4 ) { BufferLer[7]= '4'; }
		if( ( aux-parametro/10 ) == 5 ) { BufferLer[7]= '5'; }
		if( ( aux-parametro/10 ) == 6 ) { BufferLer[7]= '6'; }
		if( ( aux-parametro/10 ) == 7 ) { BufferLer[7]= '7'; }
		if( ( aux-parametro/10 ) == 8 ) { BufferLer[7]= '8'; }
		if( ( aux-parametro/10 ) == 9 ) { BufferLer[7]= '9'; }
		if( ( aux-parametro%10 ) == 0 ) { BufferLer[8]= '0'; }
		if( ( aux-parametro%10 ) == 1 ) { BufferLer[8]= '1'; }
		if( ( aux-parametro%10 ) == 2 ) { BufferLer[8]= '2'; }
		if( ( aux-parametro%10 ) == 3 ) { BufferLer[8]= '3'; }
		if( ( aux-parametro%10 ) == 4 ) { BufferLer[8]= '4'; }
		if( ( aux-parametro%10 ) == 5 ) { BufferLer[8]= '5'; }
		if( ( aux-parametro%10 ) == 6 ) { BufferLer[8]= '6'; }
		if( ( aux-parametro%10 ) == 7 ) { BufferLer[8]= '7'; }
		if( ( aux-parametro%10 ) == 8 ) { BufferLer[8]= '8'; }
		if( ( aux-parametro%10 ) == 9 ) { BufferLer[8]= '9'; }
	}
	return 0;
}
```

```
int Funcao_Identifica_Parametro_Escrita (int parametro)
{
	int aux=0;
	BufferEscrever[0] = 0x04; BufferEscrever[1] = 0x41; BufferEscrever[2] = 0x02;
	BufferEscrever[5] = 0x00; BufferEscrever[7] = 0x08; BufferEscrever[10] = 0x3D;
	BufferEscrever[11] = 0x00; BufferEscrever[12] = 0x00; BufferEscrever[13] = 0x00;
	BufferEscrever[14] = 0x00; BufferEscrever[15] = 0x03; BufferEscrever[16] = 0x00;

	if ((parametro<100))			{	BufferEscrever[6] = '1'; aux =   0; }
	if ((parametro>=100) && (parametro<=199)) { BufferEscrever[6] = '2'; aux = 100; }
	if ((parametro>=200) && (parametro<=299)) { BufferEscrever[6] = '3'; aux = 200; }
	if ((parametro>=300) && (parametro<=399)) { BufferEscrever[6] = '4'; aux = 300; }
	if ((parametro>=400) && (parametro<=499)) { BufferEscrever[6] = '5'; aux = 400; }
	if ((parametro>=500) && (parametro<=599)) { BufferEscrever[6] = '6'; aux = 500; }
	if ((parametro>=600) && (parametro<=699)) { BufferEscrever[6] = '7'; aux = 600; }

	if(aux == 0)
	{
		if( ( (parametro)/10 ) == 0 ) { BufferEscrever[8]= '0'; }
		if( ( (parametro)/10 ) == 1 ) { BufferEscrever[8]= '1'; }
		if( ( (parametro)/10 ) == 2 ) { BufferEscrever[8]= '2'; }
		if( ( (parametro)/10 ) == 3 ) { BufferEscrever[8]= '3'; }
		if( ( (parametro)/10 ) == 4 ) { BufferEscrever[8]= '4'; }
		if( ( (parametro)/10 ) == 5 ) { BufferEscrever[8]= '5'; }
		if( ( (parametro)/10 ) == 6 ) { BufferEscrever[8]= '6'; }
		if( ( (parametro)/10 ) == 7 ) { BufferEscrever[8]= '7'; }
		if( ( (parametro)/10 ) == 8 ) { BufferEscrever[8]= '8'; }
		if( ( (parametro)/10 ) == 9 ) { BufferEscrever[8]= '9'; }
		if( ( (parametro)%10 ) == 0 ) { BufferEscrever[9]= '0'; }
		if( ( (parametro)%10 ) == 1 ) { BufferEscrever[9]= '1'; }
		if( ( (parametro)%10 ) == 2 ) { BufferEscrever[9]= '2'; }
		if( ( (parametro)%10 ) == 3 ) { BufferEscrever[9]= '3'; }
		if( ( (parametro)%10 ) == 4 ) { BufferEscrever[9]= '4'; }
		if( ( (parametro)%10 ) == 5 ) { BufferEscrever[9]= '5'; }
		if( ( (parametro)%10 ) == 6 ) { BufferEscrever[9]= '6'; }
		if( ( (parametro)%10 ) == 7 ) { BufferEscrever[9]= '7'; }
		if( ( (parametro)%10 ) == 8 ) { BufferEscrever[9]= '8'; }
		if( ( (parametro)%10 ) == 9 ) { BufferEscrever[9]= '9'; }
	}
	 else
	{
		if( ( aux-parametro/10 ) == 0 ) { BufferEscrever[8]= '0'; }
		if( ( aux-parametro/10 ) == 1 ) { BufferEscrever[8]= '1'; }
		if( ( aux-parametro/10 ) == 2 ) { BufferEscrever[8]= '2'; }
		if( ( aux-parametro/10 ) == 3 ) { BufferEscrever[8]= '3'; }
		if( ( aux-parametro/10 ) == 4 ) { BufferEscrever[8]= '4'; }
		if( ( aux-parametro/10 ) == 5 ) { BufferEscrever[8]= '5'; }
		if( ( aux-parametro/10 ) == 6 ) { BufferEscrever[8]= '6'; }
		if( ( aux-parametro/10 ) == 7 ) { BufferEscrever[8]= '7'; }
		if( ( aux-parametro/10 ) == 8 ) { BufferEscrever[8]= '8'; }
		if( ( aux-parametro/10 ) == 9 ) { BufferEscrever[8]= '9'; }
		if( ( aux-parametro%10 ) == 0 ) { BufferEscrever[9]= '0'; }
		if( ( aux-parametro%10 ) == 1 ) { BufferEscrever[9]= '1'; }
		if( ( aux-parametro%10 ) == 2 ) { BufferEscrever[9]= '2'; }
		if( ( aux-parametro%10 ) == 3 ) { BufferEscrever[9]= '3'; }
		if( ( aux-parametro%10 ) == 4 ) { BufferEscrever[9]= '4'; }
		if( ( aux-parametro%10 ) == 5 ) { BufferEscrever[9]= '5'; }
		if( ( aux-parametro%10 ) == 6 ) { BufferEscrever[9]= '6'; }
		if( ( aux-parametro%10 ) == 7 ) { BufferEscrever[9]= '7'; }
		if( ( aux-parametro%10 ) == 8 ) { BufferEscrever[9]= '8'; }
		if( ( aux-parametro%10 ) == 9 ) { BufferEscrever[9]= '9'; }
	}
	return 0;
}
```

## Arquivo Form.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <assert.h>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <fstream.h>
#include <stdlib.h>
#include <winNT.h>
#include <windows.h>
#include <winbase.h>
#include <dos.h>
#pragma hdrstop
#include "Form_interface.h"
#include "Funcoes.h"
#include "Form_about.h"
//---------------------------------------------------------------------
TAboutBox *AboutBox;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Chart"
#pragma link "TeEngine"
#pragma link "TeeProcs"
#pragma link "Series"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
char PortaComunica;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_conectarClick(TObject *Sender)
{
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        if (retorno_conecta != 0)
        {
                ShowMessage("Erro na conexão");
        }
        else
        {
                ShowMessage("Conexão estabelecida");
                retorno_baud_rate=Cria_BaudRate ();
                if (retorno_baud_rate==0)
                {
                        ShowMessage("Bauld Rate ok");
                        bt_desconectar->Enabled=true;
                        bt_conectar->Enabled=false;
                        combo_com->Enabled=false;
                        bt_ler->Enabled=true;
                        bt_escrever->Enabled=true;
                }
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_desconectarClick(TObject *Sender)
{
        bt_desconectar->Enabled=false;
```

```
        bt_conectar->Enabled=true;
        combo_com->Enabled=true;
        bt_ler->Enabled=false;
        bt_escrever->Enabled=false;
        Fexa_Conexao ();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_ligaClick(TObject *Sender)
{

        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();


        int retorno_escrita;
        int TamaString;    //Para armazenar o tamanho da String.
 //     strcpy(BufferEscrever,"\x04\x41\x02""00803""\x3d\x00\x03\x00\x03\x03\x05");
        int val = 1, param = 1;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_desligaClick(TObject *Sender)
{
        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();

        int retorno_escrita;
        int TamaString;    //Para armazenar o tamanho da String.
//      strcpy(BufferEscrever,"\x04\x41\x02""00803""\x3d\x00\x01\x00\x00\x03\x04");
        int val = 0, param = 1;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_sentido_horarioClick(TObject *Sender)
{
        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();

        int retorno_escrita;
        int TamaString;    //Para armazenar o tamanho da String.
//      strcpy(BufferEscrever,"\x04\x41\x02""00803""\x3d\x00\x04\x00\x04\x03\x05");
        int val = 1, param = 0;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bt_sentido_antiClick(TObject *Sender)
{
        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
```

```
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();

        int retorno_escrita;
        int TamaString;    //Para armazenar o tamanho da String.
//      strcpy(BufferEscrever,"\x04\x41\x02""00803""\x3d\x00\x04\x00\x00\x03\x01");
        int val = 0, param = 0;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::bt_escreverClick(TObject *Sender)
{
        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();

        int valparam, retorno_hexa, retorno_bcc, retorno_escrita;

        valparam = Funcao_Identifica_Parametro_Escrita (combo_escrita->ItemIndex);
//      valparam = Funcao_Identifica_Parametro_Escrita (218);
        if ( valparam == 0 ) { ShowMessage("Parametro Obtido" ); }
        if ( valparam == 1 ) { ShowMessage("Erro: Leitura do Parametro"); }

        retorno_hexa = Funcao_DecimaltoHexa (ed_valor_escrita->Text.ToInt());
        if ( retorno_hexa == 0 ) { ShowMessage("Decimal Convertido" ); }
        if ( retorno_hexa != 0 ) { ShowMessage("Erro: Conversão Decimal"); }

        retorno_bcc = Funcao_GeraBCC ();
        if ( retorno_bcc == 0 ) { ShowMessage("BCC Gerado" ); }
        if ( retorno_bcc != 0 ) { ShowMessage("Erro: Criação BCC"); }

        int TamaString;    //Para armazenar o tamanho da String.
        int val = 5, param = 5;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::bt_lerClick(TObject *Sender)
{

        Fexa_Conexao ();
        int retorno_conecta, retorno_baud_rate;
        PortaComunica = combo_com->ItemIndex;
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();

        int retorno_leitura, valparam;

        valparam = Funcao_Identifica_Parametro_Leitura ((combo_leitura->ItemIndex));
        if ( valparam == 0 ) { ShowMessage("Parametro Obtido" ); }
        if ( valparam == 1 ) { ShowMessage("Erro: Leitura do Parametro"); }

        retorno_leitura = Funcao_Leitura ();
        if ( retorno_leitura == 0 ) { ShowMessage("Valor Lido" ); }
        if ( retorno_leitura == 1 ) { ShowMessage("Valor não lido"); }
```

```
		ed_retorno_leitura->Text = Funcao_HexatoDecimal_Leitura ();

	//Agora tem que ler a variavel BufferLerResposta converter para Decimal

	}
	//---------------------------------------------------------------------------

	void __fastcall TForm1::TrackBar1Change(TObject *Sender)
	{
	te_velocidade->Text=TrackBar1->Position;
	}
	//---------------------------------------------------------------------------

	void __fastcall TForm1::te_velocidadeChange(TObject *Sender)
	{
		if (te_velocidade->Text=="-")
		{
	//			ShowMessage("Erro, caractere não permitido '-' ");
				TrackBar1->Position=0;
				te_velocidade->Text="0";
		}
		if (te_velocidade->Text=="")
		{
				TrackBar1->Position=0;
		}
		else
		{
				if(StrToInt(te_velocidade->Text)>1800)
				{
					TrackBar1->Position=1800;
					te_velocidade->Text="1800";
				}
				else
				{
					if (StrToInt(te_velocidade->Text)<-0)
					{
						TrackBar1->Position=0;
						te_velocidade->Text="0";
					}
					else
					{
						if((StrToInt(te_velocidade->Text)>=0)&&(StrToInt(te_velocidade-
>Text)<=1800))
						{
							TrackBar1->Position=StrToInt(te_velocidade->Text);
						}
						else
						{
	//						ShowMessage("Erro");
							TrackBar1->Position=0;
							te_velocidade->Text="0";
						}
					}
				}
		}
	}
	//---------------------------------------------------------------------------

	void __fastcall TForm1::bt_set_velocidadeClick(TObject *Sender)
	{

		Fexa_Conexao ();
		int retorno_conecta, retorno_baud_rate;
		PortaComunica = combo_com->ItemIndex;
```

```
        retorno_conecta=Cria_Conexao(PortaComunica);
        retorno_baud_rate=Cria_BaudRate ();


        int retorno_escrita;
        int TamaString;    //Para armazenar o tamanho da String.

        int valparam, retorno_hexa, retorno_bcc;

        retorno_hexa = Funcao_DecimaltoHexa (1+ (te_velocidade->Text.ToInt()*4.550556));
        if ( retorno_hexa == 0 ) { ShowMessage("Decimal Convertido" ); }
        if ( retorno_hexa != 0 ) { ShowMessage("Erro: Conversão Decimal"); }

        retorno_bcc = Funcao_GeraBCC ();
        if ( retorno_bcc == 0 ) { ShowMessage("BCC Gerado" ); }
        if ( retorno_bcc != 0 ) { ShowMessage("Erro: Criação BCC"); }

        int val = 5, param = 5;
        retorno_escrita = Funcao_Escrita (val,param);
        if ( retorno_escrita == 0 ) { ShowMessage("Escrita Feita" ); }
        if ( retorno_escrita == 1 ) { ShowMessage("Erro: Escrita"); }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::AboutClick(TObject *Sender)
{
AboutBox->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SairClick(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

# APÊNDICE C – Artigo aceito pela ICECE 2011

# DESENVOLVIMENTO DE SOFTWARE COM PROPÓSITO EDUCACIONAL PARA PARAMETRIZAÇÃO, MONITORAMENTO E SUPERVISÃO DE CONVERSORES DE FREQUÊNCIA

*Marcelo Christiano da França Júnior[1], Valmir Machado Pereira[2], Edson Antonio Batista[3]*

Resumo - *A necessidade de profissionais no sistema produtivo, com conhecimentos em processos industriais, capazes de compreender as interações entre os sistemas de controle e o processo é cada vez maior. O sistema de supervisão ou software supervisório é fundamental para automação de processos, pois permite aos operadores uma visão em tempo real do funcionamento da planta industrial, podendo ser operado remotamente o que favorece maior agilidade e segurança. Com base neste cenário foi desenvolvido um software supervisório para monitorar e atuar na operação de um motor de indução trifásico com carga variável no eixo. Com isto desenvolveu-se um software capaz de controlar e atuar em diferentes processos que utilize a porta serial e os mesmos protocolos de comunicação adotados pelo programa. Este software possui recursos para ser utilizado em ambiente acadêmico e industrial.*

*Palavras-chave – software supervisório, sistemas supervisórios, protocolos de comunicação.*

*Abstract - The need for professionals in the production system, with expertise in industrial processes, capable of understanding the interactions between control systems and the process is growing. The system of supervision or monitoring software is key to process automation, allowing operators to view a real-time operation of the plant and can be operated remotely that favors agility and safety. Based on this scenario was developed a monitoring software to monitor and act on the operation of an induction motor with variable load on the shaft. With it has developed a software that can monitor and act on different processes that use the serial port and the same communication protocols adopted by the program. This software has the resources to be used in academia and industry.*

*Keywords - monitoring software, supervisory systems, communication protocols.*

## INTRODUÇÃO

Na automação industrial os computadores assumem um papel fundamental na aquisição, distribuição e controle das informações. Com isto os sistemas de monitoramento e controle tornam-se essenciais, em uma linha de produção [1].

No monitoramento de uma planta industrial pode-se notar a diferença entre um sistema usando uma interface homem máquina (IHM) ou um software supervisório. A automação industrial utilizando um software supervisório apresenta diversos recursos que os IHMs não possuem, como por exemplo, geração de relatório, banco de dados, além de maiores capacidades de processamento [1].

Além da facilidade de relatórios e confiabilidade dos dados obtidos os softwares supervisório podem ser instalados em locais diferentes de onde se encontra o dispositivo controlado, podendo-se assim controlar toda uma planta de uma indústria apenas em um local desde que este contenha canais de comunicação com todos os equipamentos que serão monitorados e/ou controlados [1].

Assim é essencial que as escolas de formação de profissionais adéqüem-se e treinem nestes tipos de aplicações industriais.

Algumas empresas já fornecem software de monitoramento e atuação via software, contudo geralmente são softwares com funções específicas deixando assim o usuário restrito apenas as funcionalidades disponibilizadas pelo fabricante.

Deste modo este artigo demonstra o desenvolvimento de um software capaz de monitorar, atuar e controlar equipamentos industriais, usando como base para programar um inversor de freqüência, com uma interface amigável e de fácil entendimento pelos usuários.

Para atingir este objetivo foram estudados os sistemas operacionais comuns em aplicações industriais, em seguida foram pesquisadas as linguagens de programação em conformidade com o sistema operacional selecionado. Por fim estudou-se o conversor de freqüência para entendimento do funcionamento de sua comunicação.

## SISTEMAS OPERACIONAIS

Uma vez definido que os computadores de mesa seriam utilizados como plataforma para desenvolvimento do software proposto, foi necessário identificar os principais sistemas operacionais existentes e analisar qual seria mais apto para o projeto.

Um sistema operacional (S.O.) deve gerenciar os componentes físicos de um computador e fornecer aos programas do usuário uma interface com o hardware mais simples [1].

Atualmente, existem três sistemas operacionais comuns em computadores de mesa (*desktop*): o *Windows*, o Mac e o

1 Marcelo Christiano da França Júnior, Mestrando pela Universidade Federal de Mato Grosso do Sul, Departamento de engenharia elétrica, Campo Grande,-MS Brasil, marcelo_christiano@hotmail.com

2 Valmir Machado Pereira, Professor doutor da Universidade Federal de Mato Grosso do Sul, Departamento de Engenharia Elétrica, Campo Grande, -MS, Brasil, valmir@del.ufms.br

3 Edson Antonio Batista, Professor doutor da Universidade Federal de Mato Grosso do Sul, Departamento de Engenharia Elétrica, Campo Grande, -MS, Brasil, edson@del.ufms.br

Linux [2]. Cada um com sua característica como descrito a seguir.

- *Windows:* é o mais popular entre eles é utilizado em nove de cada dez microcomputadores, possui a desvantagem de ser pago, é desenvolvido pela Microsoft.
- Linux: sistema operacional derivado do antigo UNIX, sistema confiável e possui a grande vantagem de ser gratuito, desenvolvido na base do companheirismo por programadores do mundo todo e não é uma opção prática para o uso doméstico nos computadores de usuários finais, pois seu manuseio não é muito amigável.
- Mac OS: desenvolvido pela Apple Macintosh, tão amigável quanto o *Windows*, mas apenas pode ser utilizado em computadores Apple Macintosh diferente do Linux e *Windows* que são utilizados em computadores pessoais e não há muitos programas em português [2,3,4,5].

Existem ainda outros sistemas operacionais além dos citados, mas não são tão populares ou utilizados no mercado [2].

O Windows é um sistema operacional popular na indústria e em computadores domésticos. O Linux é um sistema operacional flexível, facilitando a configuração conforme a aplicação. O Mac Os vem sendo utilizado principalmente em máquinas específicas principalmente no ramo de edição gráficas devido as suas características próprias desenvolvidas pela Apple Macintosh.

## PROTOCOLO DE COMUNICAÇÃO

Para o acesso direto a porta serial se faz necessário o conhecimento das API's (*Application Programming Interface* - Interface de Programação de Aplicações) do *Windows*, estas API's são pequenos programas dentro do sistema operacional que auxiliam na execução de várias atividades dentro do sistema operacional [8].

A Microsoft, desenvolvedora do Windows disponibiliza na internet as bibliotecas disponíveis assim como exemplos de configurações e uso, facilitando a comunicação entre os programas desenvolvidos por usuários e o sistema operacional.

Algumas funções prontas foram utilizadas para criar uma conexão entre o software e o sistema operacional podendo se destacar as funções:

- ***CreateFile* ( )**: Função responsável pela abertura de um canal de comunicação entre o computador e a porta serial liberando assim suas funções perante ao S.O.
- ***CloseHandle* ( )**: Função responsável pelo fechamento da conexão entre o computador e a porta serial do S.O.
- ***BuildCommDCB* ( )**: Função responsável pelo preenchimento dos campos da estrutura DCB com valores especificados numa string de controle de dispositivos.
- ***ReadFile* ( )**: Função de leitura de dados via porta serial, esta função é utilizada toda vez que precisa-se ler algo que esteja no barramento serial.
- ***WriteFile* ( )**: Função de escrita de dados via porta serial, esta função é utilizada toda vez que precisa-se escrever algo no barramento serial.
- ***GetCommState* ( )** // ***SetCommState* ( )**: Estas funções respectivamente tratam de ler e alterar informações referentes ao parâmetro de comunicação serial como taxa de transmissão, paridade, entre outros.
- ***GetCommTimeouts*( )** // ***SetCommTimeouts*( )**: Estas funções são responsáveis respectivamente por ler e alterar as informações de Timeouts dependendo do protocolo utilizado pode-se alterar o tempo de espera de leitura de um bit na porta serial ou verificar se foi passado este tempo.

## LINGUAGENS DE PROGRAMAÇÃO

Para a criação de um sistema de monitoramento existem várias linguagens de programação no mercado, com isto é necessário um estudo prévio e comparação entre algumas mais conhecidas, a fim de escolher a que melhor se adequasse ao projeto proposto.

Para comparação foram utilizadas as linguagens JAVA e C++, sendo uma comparação detalhada destas linguagens

A linguagem Java se tornou a linguagem preferida para programar aplicativos baseados em Intranet e Internet e softwares para dispositivos que se comunicam através de uma rede, outro fator atrativo é que a tecnologia JAVA é totalmente orientada a objetos [9].

A linguagem de programação C++ foi desenvolvida para propósito geral com suporte para diversas aplicações [10].

Uma quantidade enorme de código C foi escrita nas empresas nas últimas décadas. C++ por ser uma evolução da linguagem C Os programadores podem executar código C, compilá-lo em um compilador C++, freqüentemente com alterações apenas na nomenclatura das palavras reservadas [9].

Ambas as tecnologias possuem uma grande quantidade de documentação disponível na internet assim como livros, além de diversos ambientes de desenvolvimentos que auxiliam na programação.

O que levou a escolha do C++ como linguagem a ser utilizada foi à facilidade de integração entre os protocolos de comunicação com o sistema operacional já que o *Windows* possui bibliotecas escritas em C facilitando assim todo processo de desenvolvimento do sistema.

## INVERSOR DE FREQÜÊNCIA

De acordo com o manual do usuário do fabricante [11], o este dispositivo é destinado para o acionamento de motores trifásicos com potência de 1,5 a 1500 cavalos vapor. Este inversor tem incorporado as mais avançadas tecnologias disponíveis mundialmente para variação de velocidade em motores de indução trifásicos.

Ainda segundo o manual as principais aplicações do inversor se dão principalmente em bombas centrífugas, bombas dosadoras de processo, ventiladores, exaustores, agitadores, misturadores, extrusoras, esteiras transportadoras, mesas de rolos, granuladores, secadores, fornos rotativos, máquinas de corte e solda, elevadores, sistemas multibombas, etc.

Entre as diversas características encontradas no inversor pode-se citar algumas que ajudaram na escolha deste modelo para desenvolvimento do projeto, são elas:

Potência variável de 1,5 a 1500 cavalos vapor tornando-se um modelo com abrangência de aplicações capaz de se adaptar a novas necessidades. Sistema de controle escalar e vetorial embutido criando possibilidades de comparação entre os tipos de controle além de PID integrado. Possui a possibilidade de comunicação entre outros dispositivos via RS – 232 ou RS – 485, facilitando o acesso às informações; Disponibilidade de 100 parâmetros configuráveis pelo usuário via software ou IHM facilitando o acesso às informações.

Utilizando a interface RS-232 tem a ligação direta entre um mestre a um inversor, ou seja, ponto a ponto, os dados podem ser transmitidos de forma bidirecional, porém não simultânea chamada de *HALF DUPLEX*. São utilizados três fios para comunicação: o (TX) para transmissão, (RX) para recepção; e (0) para retorno;

É possível fazer a supervisão do inversor utilizando a interface serial RS-232, o protocolo é baseado no tipo pergunta/resposta conforme as normas regulamentadores ISO 1745 e ISSO 646 utilizando caracteres do tipo ASCII entre inversor e o mestre. Este mestre pode ser um computador, um controlador lógico programável (CLP) entre outros. A velocidade máxima de transmissão é de 9600 bits por segundo.

Para se utilizar esta interface deve-se fazer uso do módulo RS-232 vendido separadamente pela WEG, este módulo substitui a IHM e disponibiliza uma comunicação via RS-232 utilizando o conector RJ11. Caso ainda seja necessária a utilização da HMI, o módulo RS-232 também provê a conexão para a mesma. A figura 1 mostra o módulo de comunicação RS-232.

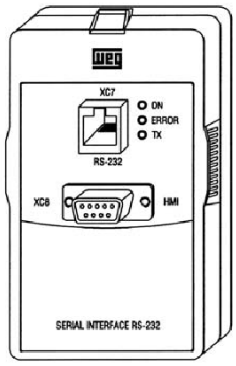


FIGURA. 1
MÓDULO DE COMUNICAÇÃO SERIAL.

O Kit de Comunicação RS-232 para PC permite a conexão do inversor a um PC através da interface RS-232 e é composto de um módulo RS-232 *Serial Interface*, cabo 3 metros RJ–11 para DB9 e o *software* SuperDrive para *Windows* que permite a programação, operação e monitoração do inversor.

Para instalação do kit de comunicação RS-232 deve-se retirar a IHM do inversor, instalar o módulo RS-232 no lugar da IHM, instalar o software SuperDrive no computador e por fim conectar o PC ao inversor usando o cabo conector, sempre deve-se seguir as instruções de operação do software SuperDrive.

O protocolo de transmissão segue a norma ISO 1745 para transmissão de dados em código. São usadas somente seqüências de caracteres de texto sem cabeçalho. A monitoração dos erros é feita através de transmissão relacionada à paridade dos caracteres individuais de 7 bits, conforme ISO 646. A monitoração de paridade é feita conforme DIN 66219 (paridade par).

São usados dois tipos de mensagens (pelo mestre): Telegrama de leitura: para consulta do conteúdo das variáveis dos inversores; Telegrama de escrita: para alterar o conteúdo das variáveis ou enviar comandos para os inversores.

## DESENVOLVIMENTO DO SOFTWARE

Para programação do protocolo foi necessário o conhecimento de algumas bibliotecas dentro do ambiente de desenvolvimento utilizado, dentre elas pode-se citar a biblioteca chamada *Windows.h* é responsável por todos os parâmetros que podem ser ligados a este sistema operacional. A partir desta biblioteca podem-se utilizar os parâmetros de abertura de um canal de comunicação com a porta serial. Na figura 2, apresenta-se a criação de uma nova conexão.

```
hCom = CreateFile   (
NomePorta,   //Nome da porta.
GENERIC_READ | GENERIC_WRITE,   //Para leitura e escrita.
0,       //Nenhuma outra abertura será permitida.
NULL,  //Atributos de segurança. (NULL) padrão.
OPEN_EXISTING,  //Criação ou abertura.
0, //Entrada e saida sem overlapped.
NULL     //Atributos e Flags. Deve ser NULL para COM.
);
```

FIGURA 2

FUNÇÃO RESPONSÁVEL PELA CRIAÇÃO DA CONEXÃO COM A PORTA SERIAL.

Após a criação de uma nova conexão é feita uma comparação com o comando *INVALID_HANDLE_VALUE*, se esta comparação for verdadeira houve algum erro na criação do canal de comunicação, caso contrário, significa que a comunicação foi estabelecida. Esta comparação é mostrada na figura 3.

```
if(hCom == INVALID_HANDLE_VALUE)
{
      return 1;    //Erro ao tentar abrir a porta especificada.
}
```

FIGURA 3

FUNÇÃO DE VALIDAÇÃO DE CRIAÇÃO DA COMUNICAÇÃO COM A PORTA SERIAL.

Depois de concluída esta etapa de criação de uma nova conexão é necessário à configuração dos parâmetros de comunicação como velocidade de transmissão, quantidade de caracteres, paridade e quantidade de bits de parada, para isto foi utilizada a função descrita na figura 4.

```
DCB dcb; //Estrutura DCB é utilizada para definir todos os parâmetros da comunicação.
dcb.BaudRate = CBR_9600;
dcb.ByteSize = 7;
dcb.Parity = EVENPARITY;
dcb.StopBits = ONESTOPBIT;

//Define novo estado.
if( SetCommState(hCom, &dcb) == 0 )
{
    return 1;   //Erro.
}
```

FIGURA 4

CONFIGURAÇÃO DCB UTILIZADA PARA DEFINIR OS PARÂMETROS DE COMUNICAÇÃO.

O protocolo escolhido foi de 9600bps com 7 bytes com paridade par e 1 bit de parada, mas para outros projetos existem outras possibilidades que podem ser acompanhadas pela tabela 1, ficando ao programador a escolha conforme as especificações do projeto.

TABELA 1

POSSIBILIDADES DE CONFIGURAÇÃO DE TRANSMISSÃO.

| *BaudRate* (Velocidade) | | *ByteSize* (Tamanho dos dados) | *Parity* (Paridade) | *StopBits* (Bits de parada) |
|---|---|---|---|---|
| CBR_110 | CBR_19200 | 5 | *EVENPARITY* (par) | *ONESTOPBIT* (1) |
| CBR_300 | CBR_38400 | 6 | *MARKPARITY* (marca) | *ONE5STOPBITS* (1.5) |
| CBR_600 | CBR_56000 | 7 | *NOPARITY* (nenhuma) | *TWOSTOPBITS* (2) |
| CBR_1200 | CBR_57600 | 8 | *ODDPARITY* (ímpar) | |
| CBR_2400 | CBR_115200 | | | |
| CBR_4800 | CBR_128000 | | | |
| CBR_9600 | CBR_256000 | | | |
| CBR_14400 | | | | |

Na figura 5 mostra-se outra função de comunicação, esta função é responsável por liberar o sistema de sua comunicação. Não executar esta função pode gerar muitos problemas, pois novas conexões ficam bloqueadas até o sistema todo ser reiniciado manualmente.

```
int Fexa_Conexao (HANDLE hCom)
{
    CloseHandle( hCom );  //Fecha a porta aberta anteriormente por CreateFile().
    return 0;
}
```

FIGURA 5

FUNÇÃO RESPONSÁVEL PELO ENCERRAMENTO DA CONEXÃO.

Nas figuras 6 e 7 são demonstradas as rotinas de leitura e de escrita de dados respectivamente, pode-se notar um comando *sleep* este comando é necessário para o correto recebimento dos dados pois se a leitura for feita antes do envio da resposta, o programa pode adquirir dados anteriores que ainda estavam no barramento.

```
BytesEscritos = 0;
BytesLidos = 0;
WriteFile( hCom, BufferLer, TamaString, &BytesEscritos, NULL );
Sleep(15);
ReadFile( hCom, BufferRecebe, 14, &BytesLidos, NULL );
```

FIGURA 6

CÓDIGO DE TELEGRAMA DE LEITURA.

```
BytesEscritos = 0;
BytesLidos = 0;
WriteFile( hCom, BufferLer, TamaString, &BytesEscritos, NULL );
Sleep(15);
ReadFile( hCom, BufferRecebe, 2, &BytesLidos, NULL );
```

FIGURA 7

CÓDIGO DO TELEGRAMA DE ESCRITA.

Com o decorrer do projeto outras funções foram criadas para auxiliar o software tanto na parte sistêmica do projeto quanto para o entendimento do usuário, as funções de conversões de números decimais para números binários e vice e versa, o gerador de bit de verificação ou *byte* de *CheCksum* foram de total importância para o projeto, uma vez que o sistema entende apenas números em hexadecimal e o usuário padrão utiliza o sistema decimal e finalmente para validar o recebimento e envio de dados utilizou-se a função de gerador de bit de verificação.

## INTERFACE

Após estabelecida a comunicação e configuração dos parâmetros, desenvolveu-se as interfaces que interagem com a automação proposta neste projeto.

Foi proposto ao trabalho gerar duas telas de monitoramento, a primeira ilustrada na figura 8 mostra a primeira interface para o usuário, esta criada para mostrar o que poderia ser executado no sistema como a configuração da comunicação, leitura e escrita de parâmetros Além de controle de velocidade e sentido de giro do motor.

FIGURA 8
PRIMEIRA INTERFACE PARA MONITORAMENTO.

Na figura 9 são mostrados três gráficos de desempenhos de cada parâmetro escolhido pelo usuário, assim este poderá comparar visualmente o desempenho de cada ponto escolhido por ele, representada na figura 9.

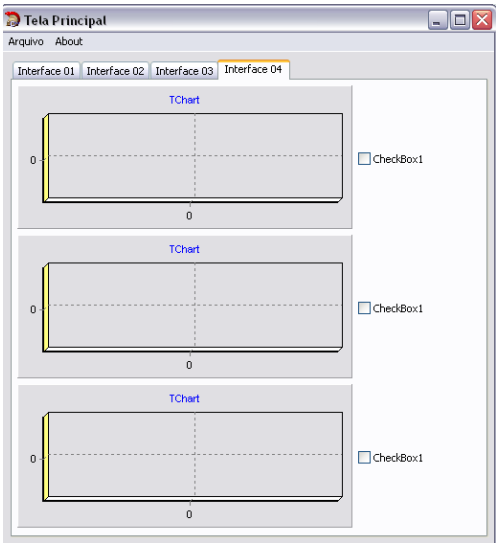


FIGURA 9
SEGUNDA INTERFACE DE MONITORAMENTO.

## CONCLUSÕES

O software desenvolvido neste projeto foi testado atuando e monitorando diversos parâmetros. Nos testes pode-se observar que além de fornecer maiores funcionalidades nas aplicações do fabricante é possível criar aplicações específicas de acordo com as necessidades, além de criar uma interface de mais fácil entendimento para o usuário.

As sub-rotinas criadas com as funções do projeto se tornam um referencial para novos projetos devido as suas estruturas, sendo facilmente adaptadas as mais diversas necessidades.

## REFERÊNCIAS BIBLIOGRÁFICAS

[1] Tanenbaum, A.S. “Sistemas operacionais modernos”, Pearson Prentice Hall, 2 Ed, (2003).
[2] Silva, M.G. “Informática : Terminologia Básica : Microsoft Windows XP, Word XP, Excel XP, Access XP, PowerPoint XP”, Érica, 4 Ed, (2008).
[5] Hunt, C. “Servidores de Redes com Linux: O recurso essencial para administradores de sistemas”, Market Books, 2 Ed, (2000).
[6] Ball, B. et. al. “Dominando Red Hat Linux 7”, Ciência Moderna, 1 Ed, (2002).
[7] Pogue, D. “*Mac OS X Leopard*”, Digerati, 1 Ed, (2010).
[8] Denver. A. “*Serial Communications in Win32*”, Disponível em: <http://msdn.microsoft.com/en-us/library/ms810467#serial_topic4>. Acesso em: 14/01/2011.
[9] Deitel, “C++ Como Programar”, Prentice-Hall, 5 Ed, 2006.
[10] Stroustrup, B. “The Design and Evolution of C++”, Addison-Wesley, (1994).
[11] WEG, “Manual do inversor de freqüência CFW – 09”, Disponível em: <http://www.weg.net/files/products/WEG-cfw-09-manual-do-usuario-0899.5298-4.0x-manual-portugues-br.pdf> . Acesso em: 05/04/2010.