Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500
Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

INTEGRAÇÃO ENTRE APLICATIVOS COMERCIAIS E INDUSTRIAIS ATRAVÉS DO PADRÃO COM

Júlio César Calegari
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
julio@tronictec.com.br

José de Souza
Professor Orientador
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
Josesouza@liberato.com.br

Resumo

De modo geral os aplicativos de *software* usados na área comercial não foram desenvolvidos com o objetivo de trocar informações com dispositivos industriais. Seus recursos nativos não permitem o intercâmbio de dados na maioria dos casos. A comunicação entre diferentes tipos de aplicativos facilita a automação e controle. Este trabalho está relacionado à área de automação industrial, foi executado com o objetivo de desenvolver um aplicativo para aproveitar os recursos do Office Excel, integrando a este os protocolos de comunicação usados em redes industriais. Este aplicativo foi desenvolvido com a finalidade de realizar a conversão entre padrões de comunicação diferentes, gerenciando as informações entre dispositivos industriais e planilhas do Excel, criando uma opção simples e intuitiva para monitorar processos na indústria.

Palavras-chave: automação, supervisório, controladores lógicos programáveis.

INTEGRATION BETWEEN COMMERCIAL AND INDUSTRIAL APPLICATIONS THROUGH THE STANDARD COM

Abstract

Generally software applications used in the commercial area were not developed with the aim of exchanging information with industrial devices. His native resources do not allow the exchange of data in most cases. Communication between different types of applications facilitates the automation and control. This work is related to the area of industrial automation, was executed with the goal of developing an application to utilize the capabilities of the Office Excel, integrating this communication protocols used in industrial networks. This application was developed in order to perform the conversion between different communication standards, managing the information between industrial devices and Excel spreadsheets, creating a simple and intuitive option to monitor processes in the industry.

Key-words: *automation, supervisory, programmable logic controllers.*

1. Introdução

Nos sistemas de automação a comunicação com os mais variados tipos de dispositivos para aquisição de dados e controle de processos é muito comum. O mercado está repleto de diversas tecnologias e seus respectivos fabricantes, com diferentes protocolos de rede.

Cada fabricante possui seus próprios padrões na implementação dos protocolos de rede usados na comunicação com seus equipamentos. Segundo Bolzani (2010) atualmente existem vários protocolos de rede para a comunicação entre controladores e elementos de campo.

Para Bolzani (2010) as ferramentas existentes e comercializadas no mercado exigem do desenvolvedor uma afinidade em áreas complexas como a comunicação de dados, integrando-o com uma infinidade de protocolos usados em redes industriais. Amenizar o grau de dificuldade desta tarefa, criando uma ferramenta simples e intuitiva que disponibilize as informações sem grande esforço por parte do usuário, irá tornar este trabalho menos árduo para o desenvolvedor.

Com o intuito de aproveitar os recursos do programa Excel, busca-se integrar a este protocolos de comunicação usados em redes industriais, por meio de outro programa capaz de fazer a conversão entre padrões de comunicação diferentes, gerenciando as informações entre dispositivos industriais e planilhas do Excel. Esta ação visa construir uma alternativa para monitorar processos na indústria.

Para realizar os testes e a implementação da troca de dados entre CLP (*Programmable Logic Controller*) e a planilha do Excel será implementado o protocolo de comunicação ModbusIP. Este será disponibilizado como DLL (*Dynamic-link library*), parte de *software*, neste caso, um módulo externo com funções de acesso a dados no CLP, que irá prover suporte ao protocolo de comunicação utilizado.

Este artigo apresenta os resultados obtidos no desenvolvimento de um aplicativo capaz de interagir com protocolos usados em redes industriais, repassando a informação para *softwares* comerciais que dão suporte ao padrão COM (*Component Objet Model*).

O objetivo específico da construção de uma planilha eletrônica com uso do Office Excel, é utiliza-la como interface de visualização e operação de sistemas supervisórios. Isso não impedirá que outros programas que seguem o mesmo padrão COM para a comunicação com Excel, possam interagir como o aplicativo desenvolvido.

O Artigo é constituido da seguinte estrutura: a seção 2 apresenta o referencial teórico, a seção 3 a metodologia utilizada, a seção 4 os resultados obtidos na fase de testes do aplicativo e a seção 5 traz as conclusões sobre o aplicativo desenvovido.

2. Referencial Teórico

2.1 Controladores Programáveis

Os controladores programáveis, conhecidos como CLP, são dispositivos de *hardware* responsáveis pelo controle de processos na interação com válvulas, atuadores, solenóides e motores (BOLZANI, 2010).

Uma infinidade de equipamentos utilizados em plantas de automação na indústria são controlados via CLP, de acordo com Bolzani (2010) é um dispositivo eletrônico baseado em microprocessador programável pelo usuário, como mostra a Figura 1.

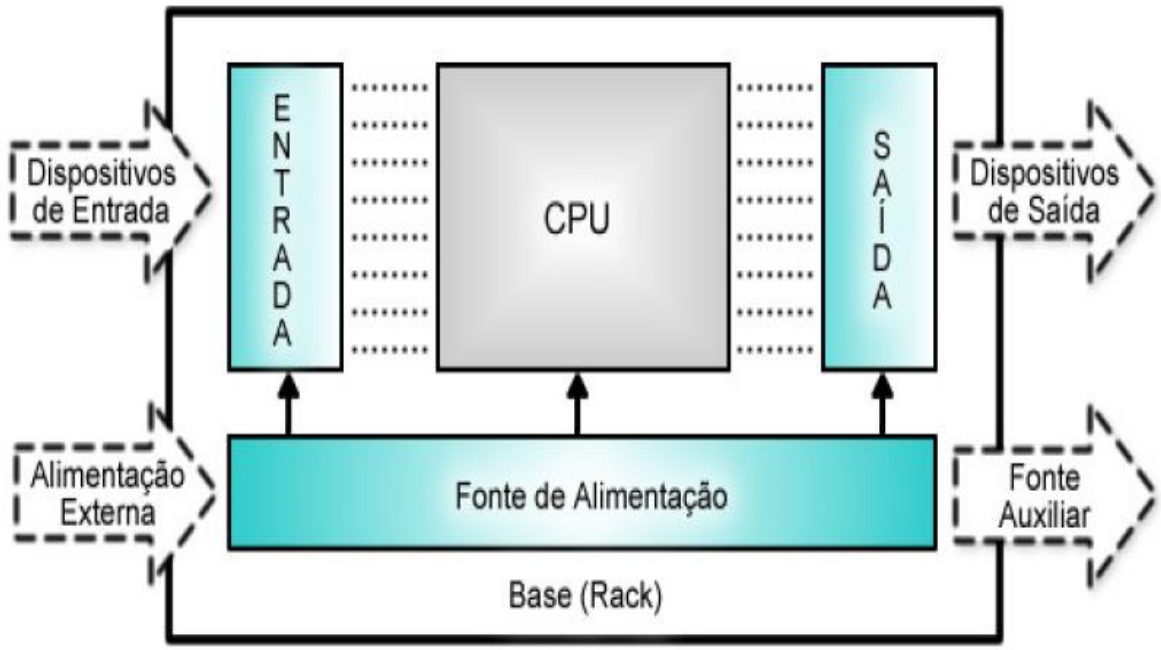

FIGURA 1 - Estrutura básica de um PLC
Adaptado de Silveira e Santos (2003)

Silva (1994) comenta que estes dispositivos são providos de memória programável para guardar instruções, lógicas de programação feitas e compiladas em linguagens computacionais desenvolvidas de acordo com as características do seu microprocessador.

Esta tecnologia foi possível graças à evolução da eletrônica, com a criação dos circuitos integrados e microprocessadores. Bolzani (2010) comenta que após o surgimento destes equipamentos no mercado ocorreu uma evolução na montagem dos painéis elétricos, permitindo uma lógica de comando flexível através da programação de CLPs, desta forma na maioria dos casos alterar as funcionalidades de um painel de comando não implica em grandes mudanças na sua montagem eletromecânica.

2.1.1 Definições sobre PLC

Em Cândido (2004) os CLPs são definidos como pequenos computadores industriais utilizados para realizar funções de controle em equipamentos na indústria. Segundo afirma o autor os dados de entrada são analizados de forma lógica por uma linguagem específica do microprocessador este por sua vez gera uma saída atuando nos equipamentos periféricos ao equipamento de PLC instalado.

Segundo a ABNT (Associação Brasileira de Normas Técnicas) o CLP é um equipamento eletrônico digital com *hardware* e *software* compatíveis com aplicações industriais (BALLOCK, 2003).

Para a NEMA (*National Electrical Manufactures Association*), é um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, controlando por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos (BALLOCK, 2003).

Seguindo estas especificações, os CLPs são equipamentos eletrônicos baseados em um microprocessador com uma memória programável que armazena instruções. Geralmente são utilizados para controle discreto, na automação de equipamentos em processos na indústria, mas como dito por Bolzani (2010) com a popularização da tecnologia estes equipamentos tem se expandido para área comercial e residêncial.

A principal característica destes dispositivos é a capacidade de progamação e reprogramação de instruções lógicas, além de ser projetado para atuar em ambiente industrial, suportando condições adversas geralmente não suportadas por computadores normais. A Figura 2 ilustra um exemplar de CLP utilizado na indústria.

FIGURA 2 - Modelo de CLP TP02 (Weg)
Fonte: (BALLOCK, 2003)

2.3 *Software* Unity

Uma ferramenta utilizada paro o desenvolvimento de aplicativos de *software* para PLCs, elaborada e desenvolvida pela empresa Schneider Eletric, esta disponível como aplicativo de configuração e programação que integra múltiplos modelos de PLCs. Estes modelos são classificados em níveis diferentes de performanse de acordo com o microprocessador utilizado em sua fabricação.

A Figura 3 mostra detalhes da interface do aplicativo Unity utilizando a linguagem Ladder (escada) de configuração.

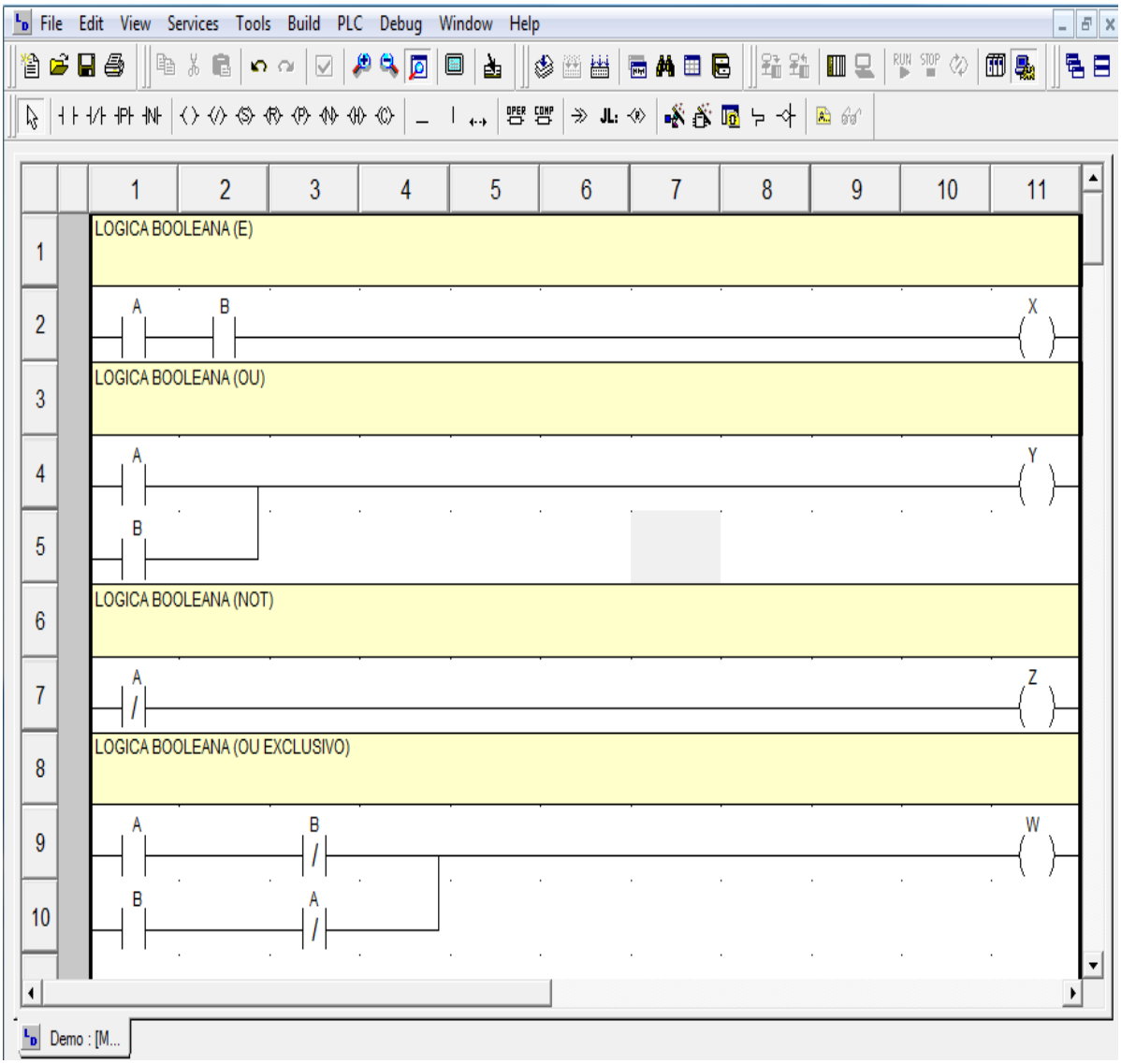

FIGURA 3 - Lógica em linguagem Ladder
Baseado em Ballock (2003)

O Ladder é uma linguagem de programação gráfica, em forma de diagrama, que representa as ligações físicas entre componentes eletrônicos (sensores e atuadores), esta linguagem nasceu na

necessidade de facilitar a programação em ambientes industriais, remetendo para uma linguagem de alto nível e facilitar a programação de PLCs para engenheiros e eletricistas (BALLOCK, 2003).

O Unity é um *software* de programação comum para as linhas de PLC Modicon Premium, Atrium e Quantum. Disponibiliza os cinco idiomas de programação exigidos pela norma IEC61131-3, que segundo Guimarães (2005) é a terceira de cinco partes que se divide a norma IEC61131, que se destinada a padronizar as linguagens de programação para CLPs.

Este *software* possui integrado em seu sistema ferramentas de depuração e diagnósticos, segundo a Schneider Eletric foi concebido para aumentar a produtividade do desenvolvimento e a facilidade de manutenção, baseado no reuso de código, através da criação de blocos de controle e bibliotecas de componentes nativas já desenvolvidas e integradas a ferramenta.

2.4 Redes Industriais

As arquiteturas de padrões em redes industriais destinam-se a integrar os componentes de um sistema de automação e são caracterizadas em função do protocolo utilizado (MARTINS, 2010).

Seguindo as teorias de Martins (2010) redes industriais são os diferentes protocolos de comunicação utilizados na troca de dados entre os equipamentos que envolvem uma planta automatizada. O desenvolvimento destas redes é baseado em modelos utilizados em barramentos de comunicação, meio físico que transmite os dados.

Modelos de redes industriais, segundo a empresa Rockwell:

- Origem Destino (ponto a ponto): Se caracterisa pela disposição em série dos integrantes da rede, fazendo com que os dados passem por todas as estações que estiverem conectadas, mas apenas a receptora poderá reconhecê-los.

A ação sincronizada entre os nós é muito difícil, uma vez que os dados chegam aos nós em momentos diferentes. Neste modelo ocorre o desperdício de recursos em função da repetição dos mesmos dados quando apenas o destino é diferente.

- Produtor Consumidor: Este modelo permite uma utilização mais eficiente da banda de comunicação, possibilitando a transmissão de grande quantidade de informação. Múltiplos nós podem simultaneamente consumir os dados de um mesmo produtor.

Este modelo é o modo mais eficiente para a transferência de dados entre vários dispositivos. Um controlador consome a variável de processo produzida pelo sensor, e produz a saída consumida pelo atuador.

São muitas as tecnologias para transmissão de dados em redes industriais, entre as mais conhecidas estão o Modbus RTU, Modbus/TCP, Profibus e DeviceNet. A grande vantagem da

utilização de redes está em ligar muitos instrumentos no campo (chão de fábrica) com uma quantidade mínima de cabos.

Entre todas as tecnologias associadas ao controle industrial, as redes de comunicação industriais são as que mais sofreram evoluções na última década (NOGUEIRA, 2009).

A partir de dois ou mais dispositivos interligados por um barramento, caracterizam uma rede onde temos que considerar sua topologia. As mais comuns são a de anel, Figura 4, barramento (*bus*) Figura 5, estrela, Figura 6, Ponto a Ponto, Figura 7 e Árvore, Figura 8, (DJIEV, 2003).

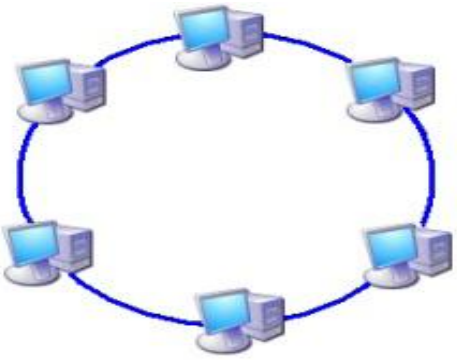

FIGURA 4 - Topologia em anel
Fonte: (NOGUEIRA, 2009)

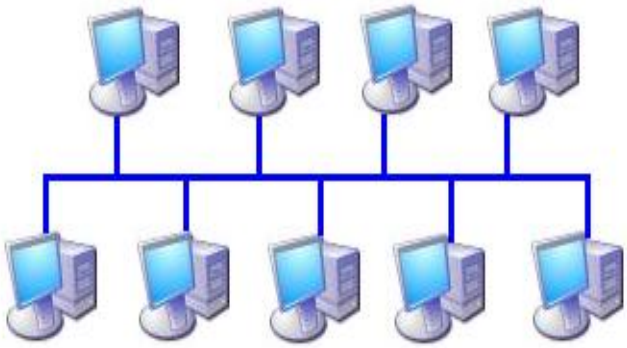

FIGURA 5 - Topologia barramento
Fonte: (NOGUEIRA, 2009)

FIGURA 6 - Topologia estrela
Fonte: (NOGUEIRA, 2009)

FIGURA 7 - Topologia ponto a ponto
Adaptado de Nogueira (2009)

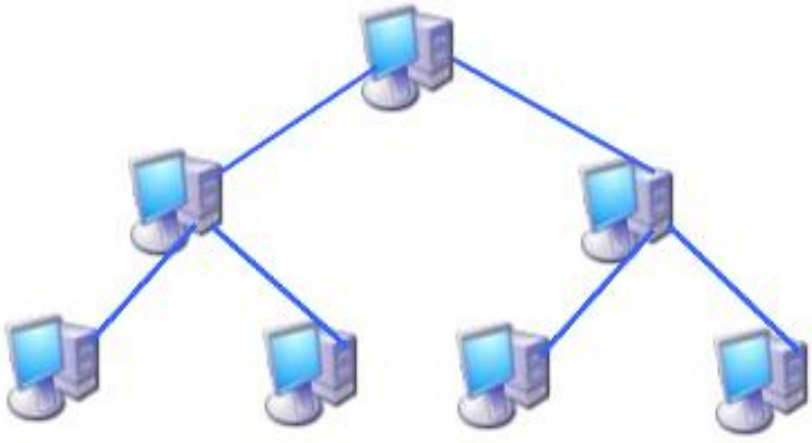

FIGURA 8 - Topologia em arvore
Adaptado de Nogueira (2009)

2.4.1 Protocolo Modbus

Criado na década de 70 para comunicação entre controladores da MODICON (Schneider) tem sua especificação aberta. Segundo Kobayashi (2009) o protocolo Modbus é baseado no paradigma de comunicação Mestre-escravo, onde o mestre é o responsável por coordenar a comunicação entre os escravos que compõem a rede.

Possui uma série de variações para diferentes tipos de barramento industrial, entre eles:

Modbus:

Utilizado em redes 232 e 485, a transmissão pode ser feita de duas maneiras, RTU ou ASCII.

- RTU: Cada byte contém dois dígitos hexadecimais.

O modo RTU transmite a informação com um menor número de bits, possui formatação de 8 bits, mas os caracteres devem ser enviados em uma sequência contínua. Este modo é o mais eficiente também é chamado de Modbus-Binário ou Modbus-B.

- ASCII: Cada byte contém um caractere ASCII entre 0 e 9, A a F.

Trabalha com formatação de 7 bits, permite intervalos de tempo maiores entre as transmissões de caracteres, chegando até um segundo. Porém sua mensagem pode chegar até o dobro do tamanho da equivalente em RTU.

ModBus/IP:

O protocolo Modbus é encapsulado no protocolo TCP/IP e transmitido através de redes padrão ethernet[1]. Trata-se de um protocolo mestre escravo quando usado no modo tradicional, porém ao ser encapsulado no TCP/IP todos os participantes da rede podem interagir simultaneamente, no sistema multi-mestre.

Durante a comunicação, a checagem de erro é feita pela conferência CRC (*Cyclic redundancy check*), segundo Oliveira (2001) a idéia básica deste algoritmo é a de simplesmente tratar a informação como um número binário e dividi -lo por um outro número binário fixo e fazer da divisão o valor de validação ou *checksum*, por segurança o mestre envia a mensagem ao escravo e aguarda um determinado tempo para gerar uma exceção de erro. Caso o escravo não responda dentro da faixa de tempo estipulada, a exceção é gerada e o erro pode ser tratado conforme o nível de sua gravidade na aplicação.

As funções requisitadas pelas mensagens de comunicação variam de acordo com o tipo de dado a ser lido na memória do CLP conforme segue:

- Função 01: efetua a leitura do estado das saídas discretas;
- Função 02: efetua a leitura do estado das entradas discretas;
- Função 03: efetua a leitura dos valores dos registradores de memória;
- Função 04: efetua a leitura dos valores das entradas analógicas;
- Função 05: efetua a escrita de uma única saída discreta;
- Função 06: efetua a escrita de um valor em um registrador de memória;
- Função 15: efetua a escrita de múltiplas saídas discretas;
- Função 16: efetua a escrita de múltiplos valores em registradores de memória.

No Quadro 1 um exemplo de Requisição Modbus do Mestre:

Endereço do Escravo	0x02	0x00	0x01	0x00	0x01	CRC
1	2	3	4	5	6	7

QUADRO 1 - Exemplo de frame Modbus Master

- Campo 1= Endereço do escravo (Interface Modbus).
- Campo 2 = Código da função.

[1] O padrão IEEE 802.3, mais conhecido como Ethernet, é uma rede de difusão de barramento com controle, descentralizado,em geral operando em velocidades de 10Mbps a10Gbps. Os computadores em uma rede Ethernet podem transmitir sempre que necessário, se dois ou mais pacotes colidirem, cada computador aguardará um tempo aleatório e fará uma nova tentativa mais tarde.

- Campos 3 e 4 = Endereço de leitura.
- Campos 5 e 6 = Quantidade de leituras .
- Campo 7 = CRC

No Quadro 2 um exemplo de Resposta Modbus do Escravo:

Endereço do Escravo	0x02	0x02	0x00	0x04	CRC
1	2	3	4	5	6

QUADRO 2 - Exemplo de frame Modbus Slave

- Campo 1 = Endereço do escravo (Interface Modbus);
- Campo 2 = Código da função;
- Campo 3 = Quantidade de bytes enviados;
- Campos 4 e 5 = Bytes de resposta;
- Campo 6 = CRC.

2.5 O Padrão COM

É um padão orientado a objetos, estabelecido pela Microsoft, define o modo como um objeto pode ser usado por outros, assim como estes podem interagir entre processos diferentes na troca de informações, ou seja, segundo Cândido (2004) é um padrão usado para que uma aplicação cliente possa acessar os serviços de uma aplicação servidora.

Esta tecnologia específica as interfaces para clientes, permitindo a comunicação padronizada. É uma coleção de métodos, funções e procedimentos relacionados que implementam os serviços disponibilizados pela interface do objeto COM (CÂNDIDO, 2004).

Através deste padrão é possível integrar aplicações distintas, desenvolvidas em diferentes plataformas, que integram à tecnologia COM. O padrão permite a comunicação entre processos paralelos, onde esta baseada a tecnologia OLE (*Object Linking and Embedding*).

De acordo com Silva (2007) o termo COM usado no desenvolvimento de ferramentas de *software*, se refere a um grupo de tecnologias que incluem OLE, OLE Automation, COM e DCOM (*Descentralized Component Object Model*).

Segundo Silva (2007) um objeto COM é um código binário, para ser utilizado por outros programas, independente da linguagem em que foram desenvolvidos.

O COM possui as seguintes funcionalidades (SILVA, 2007):

- Criar objetos que podem ser usados por várias linguagens de programação;
- Criar controles ActiveX;
- Controlar programas através da automação OLE , como Word e o Excel;
- Trabalhar com objetos ou programas em outras máquinas (DCOM).

O quadro evolutivo da tecnologia (SILVA, 2007):

- DDE (*Dynamic Data Exchange*);
- OLE;
- OLE2;
- COM;
- DCOM.

O DCOM, ideal para ser usado em sistemas distribuídos, permite a comunicação entre aplicações rodando em máquinas distintas. Segundo Cândido (2004) provê os mesmos serviços do COM, porém utiliza como meio de acesso uma rede de comunicação de dados.

2.6 Aplicativos SCADA

Os sistemas SCADA (*Supervisory Control and Data Acquisition*) são desenvolvidos para funcionar como interfaces entre o homem e a máquina (IHM). Segundo Cândido (2004) compreende um meio onde se traduz o que está acontecendo na máquina ou no sistema automatizado, ou melhor dizendo, um dispositivo que está entre o homem e a máquina.

Para Tibola (2000) supervisórios são encontrados em estações locais ou remotas de processos industriais. Estes sistemas geralmente são baseados em microcomputadores conectados a controladores programáveis (CLP).

Por meio do sistema de supervisão o operador pode interagir com o processo a ser controlado, observando, analisando e intervindo quando necessário nas variáveis de processo. O aplicativo SCADA pode atuar na coleta dados, valores dos equipamentos em campo, calculando, modificando e elaborando relatórios para futuras tomadas de decisões do operador (SILVA e SALVADOR**,** 2004).

Estes aplicativos são desenvolvidos para obter as informações de um processo, tratá-las quando necessário e disponibilizá-las ao usuário, em muitos processos o sistema de supervisão pode interagir de forma autônoma e tomar decisões pré-programadas de acordo com o cenário da aplicação em andamento (TIBOLA, 2000).

Para Silva e Salvador (2004) deve ser uma interface amigável, cujo objetivo é permitir a supervisão e o comando de determinados pontos de uma planta automatizada, recebe dados de um CLP e permite ao operador enviar sinais para que o mesmo possa mudar o comportamento dos dispositivos periféricos, atuadores, válvulas solenóides, motores, etc.

Conforme Silva e Salvador (2004) ferramentas de supervisão devem permitir que sejam monitoradas e rastreadas as informações de um processo. A interface disponibiliza um menu para o envio de comandos aos sistemas de controle (BOLZANI, 2010).

A Figura 9 mostra detalhes visuais de uma tela implementada para supervisão e controle de dispositivos.

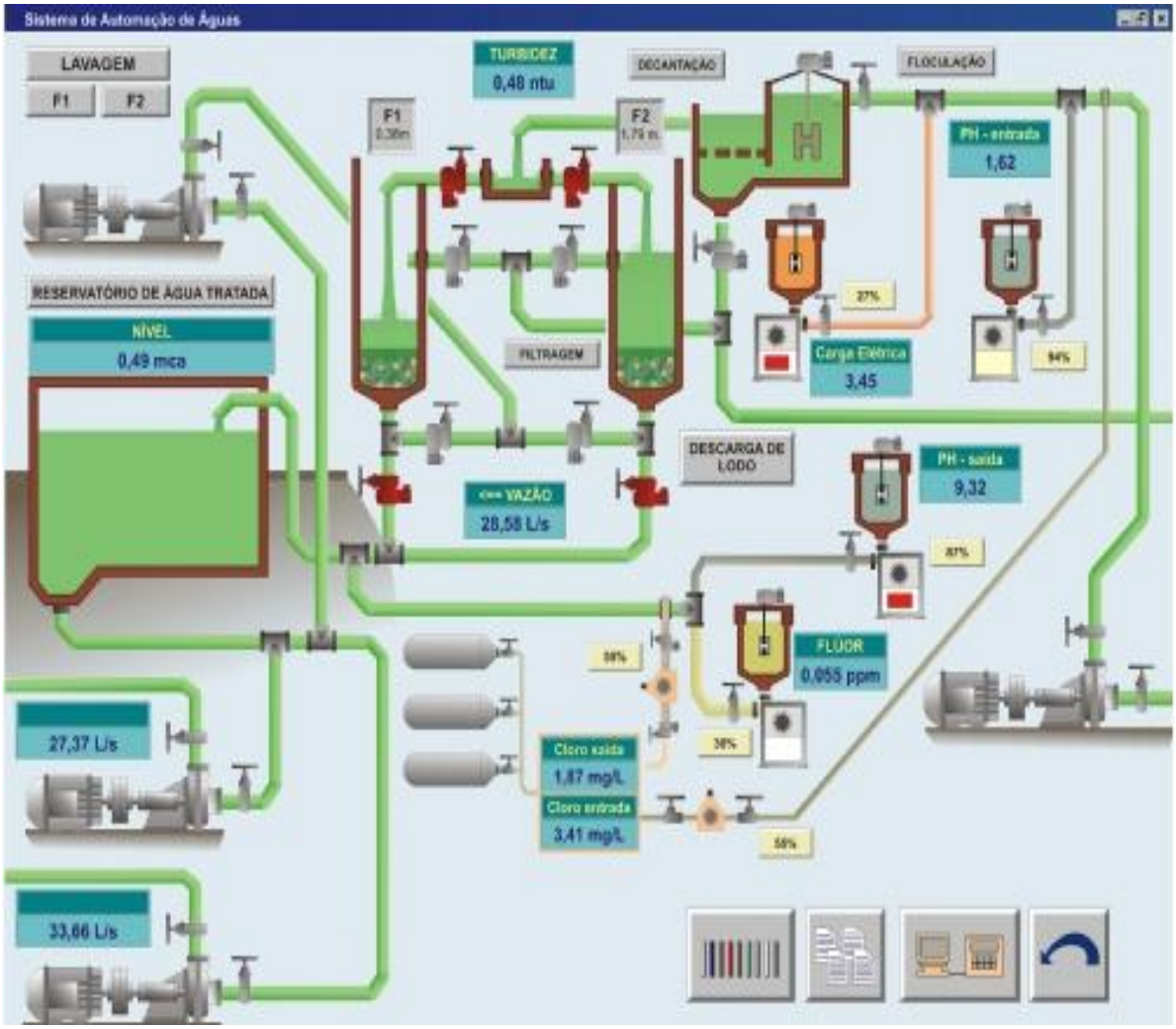

FIGURA 9 - Exemplo de interface gráfica
Baseado em Bolzani (2010)

Segundo as teorias de Bolzani (2010) os sistemas supervisórios são geralmente utilizados com a finalidade de fazer o controle de processos e o posterior reconhecimento de eventuais falhas que ocorrem em um sistema automatizado por dispositivos eletrônicos, CLPs que controlam os equipamentos através de sua interface de *hardware*.

Os aplicativos SCADA estabelecem a relação entre o homem e a máquina, para que ambos possam se comunicar através da manipulação de variáveis de processo (TIBOLA, 2000), tais variáveis são passadas entre dispositivos via rede de dados e estes interpretados deacordo com o protocolo estabelecido para comunicação entre os integrantes da rede.

A Figura 10 mostra a implentação de *hardware* de um painel elétrico de comando industrial controlando periféricos eletrônicos através de controlador lógico PLC.

FIGURA 10 - Implementação de *hardware* utilizando PLC
Baseado em Bolzani (2010)

Hoje já é comum a implementação de supervisórios nos mais diversos segmentos do comércio e da indústria, desde sistemas de alarmes, escolas, hospitais, lojas de departamento, climatização de escritórios e laboratórios, pequenas fábricas e sem falar nos inúmeros processos industriais, onde seu uso já é indispensável para a garantia de padrões de qualidade e segurança de pessoas e equipamentos.

2.7 Planilhas Excel e VBA

O sistema VBA (*Visual Basic for Applications*), recurso que permite criar funções personalizadas em planilhas do Excel, além de fazer o mesmo em outras ferramentas do pacote Office. Um complemento que permite a criação de macros, sub-rotinas para executar funções pré-programadas.

A Figura 11 mostra a inteface de codificação desta ferramenta.

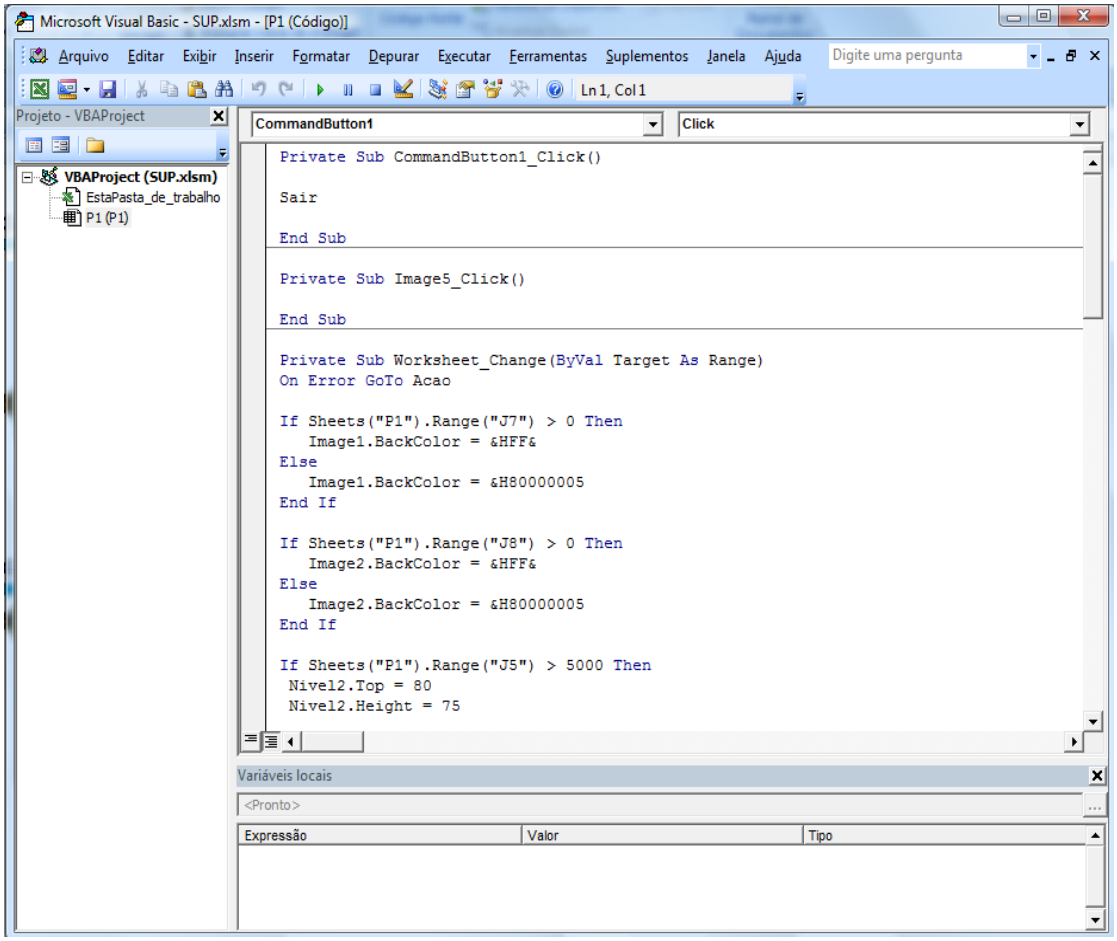

FIGURA 11 - Interface de Programação VBA

2.8 Delphi

O Delphi é um ambiente de desenvolvimento de aplicações orientado a objeto, baseado na linguagem de programação *Object* Pascal. Uma ferramenta RAD (*Rapid Application Development*), que consiste no desenvolvimento rápido de aplicações, trata-se de um modelo de desenvolvimento de *software* iterativo, que enfatiza o reuso de código, possibilitando um ciclo de desenvolvimento extremamente curto. A Figura 12 mostra a IDE (*Integrated Development Environment*) de programação.

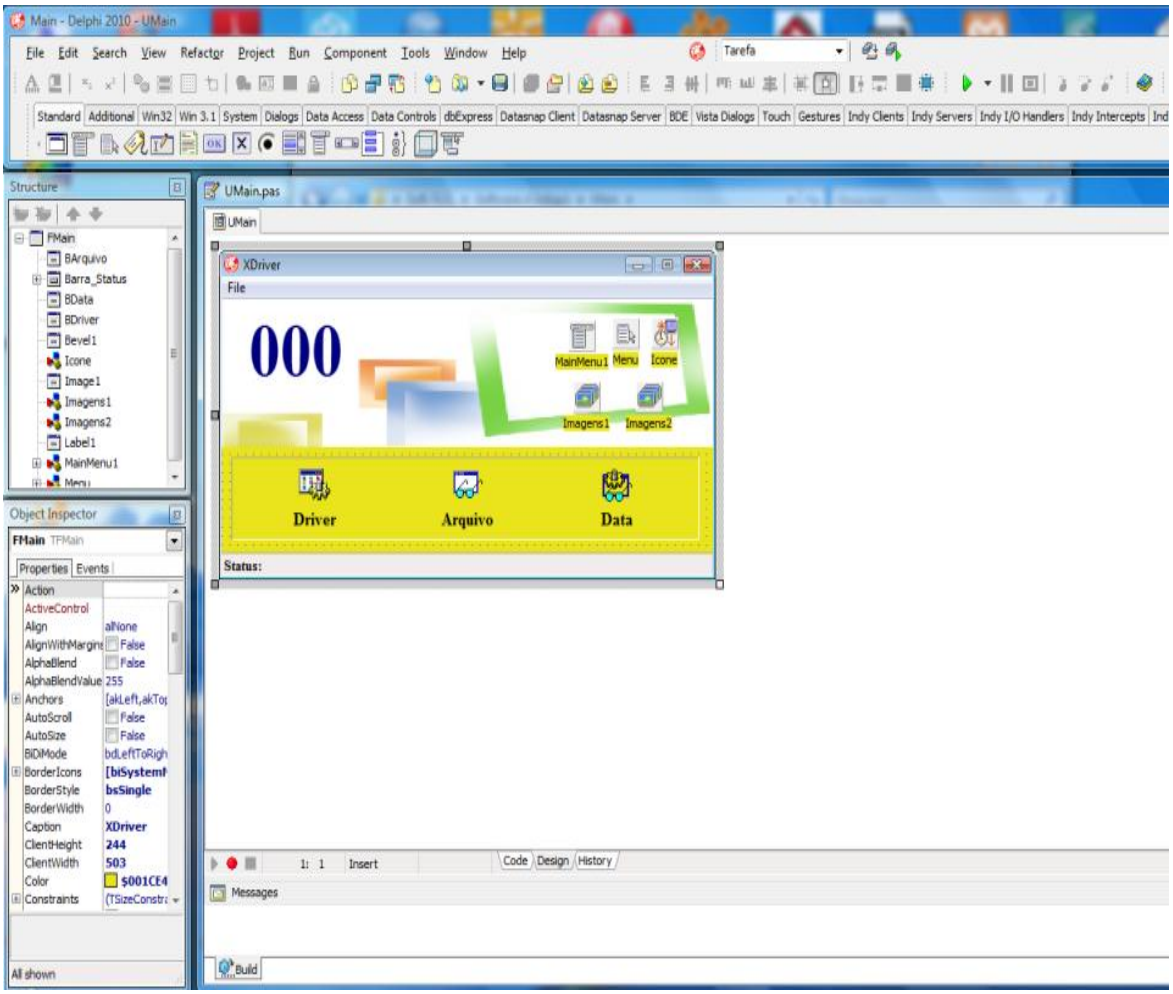

FIGURA 12 - IDE de programação Delphi

3. Metodologia

Para o desenvolvimento do sistema, o modelo adaptativo de desenvolvimento ágil DAS (*Adaptive Software Development*) foi o método eleito, este aborda os sistemas computacionais de forma simples e clara, dando mais liberdade entre as etapas do desenvolvimento (PRESSMAN, 2006).

Conforme visto em Pressman (2006) este modelo permite que etapas diferentes do projeto sejam desenvolvidas juntas em paralelo, isto ajuda na divisão das tarefas em problemas menores, muito semelhante ao conceito de orientação a objetos, onde a técnica é dividir para conquistar, além de permitir e facilitar mudanças em passos anteriores durante o desenvolvimento do projeto.

Esta metodologia foi desenvolvida em 1997 por Jim Highsmith, fundamentada na colaboração, auto-organização da equipe e aprendizado individual destes no decorrer do processo (PRESSMAN, 2006). As etapas do modelo e sua aplicação no projeto serão descritas a seguir nas etapas de especulação, colaboração e aprendizado.

3.1 Etapa de Especulação

Segundo Pressman (2006) a etapa de especulação contempla as fases de planejamento e análise do ciclo de vida do projeto, bem como o estudo do cenário da aplicação. Nesta fase foram definidos os prazos e objetivos do projeto.

Esta foi dedicada à definição e a análise das atividades de desenvolvimento do *software*. Definiu-se a divisão das partes do projeto, para que possam ser desenvolvidas paralelamente na etapa posterior.

Conforme observado por Cândido (2004) em relação sistemas OPC (*OLE for Process Control*)[2], que se assemelham a este projeto, o grande benefício é a padronização do acesso aos dados de equipamentos diversos.

Além da padronização do acesso a dados em PLCs, ocorrera a troca de informação entre aplicativos através da tecnologia COM (*Component Object Model*), uma plataforma da Microsoft para componentes de *software* lançada em 1993.

O COM é usado para permitir a comunicação entre processos e a criação dinâmica de objetos em qualquer linguagem de programação que suporte a tecnologia, para Cândido (2004) trata-se de um padrão criado pela Microsoft para que uma aplicação cliente possa acessar os serviços de uma aplicação servidora.

O objetivo definido foi criar um aplicativo de configuração fácil, amigável e intuitivo ao usuário, uma ferramenta de interface simples, permitindo que este possa se dedicar mais a parte visual do aplicativo SCADA, sem se incomodar com o gerenciamento de protocolos de comunicação e aquisição de dados em dispositivos de *hardware*. Além disso, buscou-se possibilitar que recursos visuais e funcionais de alto nível, existentes em *softwares* como o Office Excel pudessem ser aproveitados para fins de automação na área da indústria, sem a necessidade de grandes configurações por parte do usuário.

3.2 Etapa de Colaboração

Conforme Pressman (2006) a etapa de colaboração trata-se da fase de desenvolvimento simultânea de fases diferentes do projeto para serem integradas posteriormente.

[2] A OPC *Foundation* surgiu em 1995, sendo formada pelas empresas Fisher-Rosemount, Rockwell Software, Opto 22, Intellution e Intuitive Technology, com o objetivo de criar um mecanismo padronizado de troca de informações entre dispositivos dos mais diversos fabricantes.

A Figura 13 exibe o diagrama de caso de uso geral da aplicação. Este foi utilizado como referência, a base para se fazer a modelagem do *software* e atender aos requisitos funcionais do sistema.

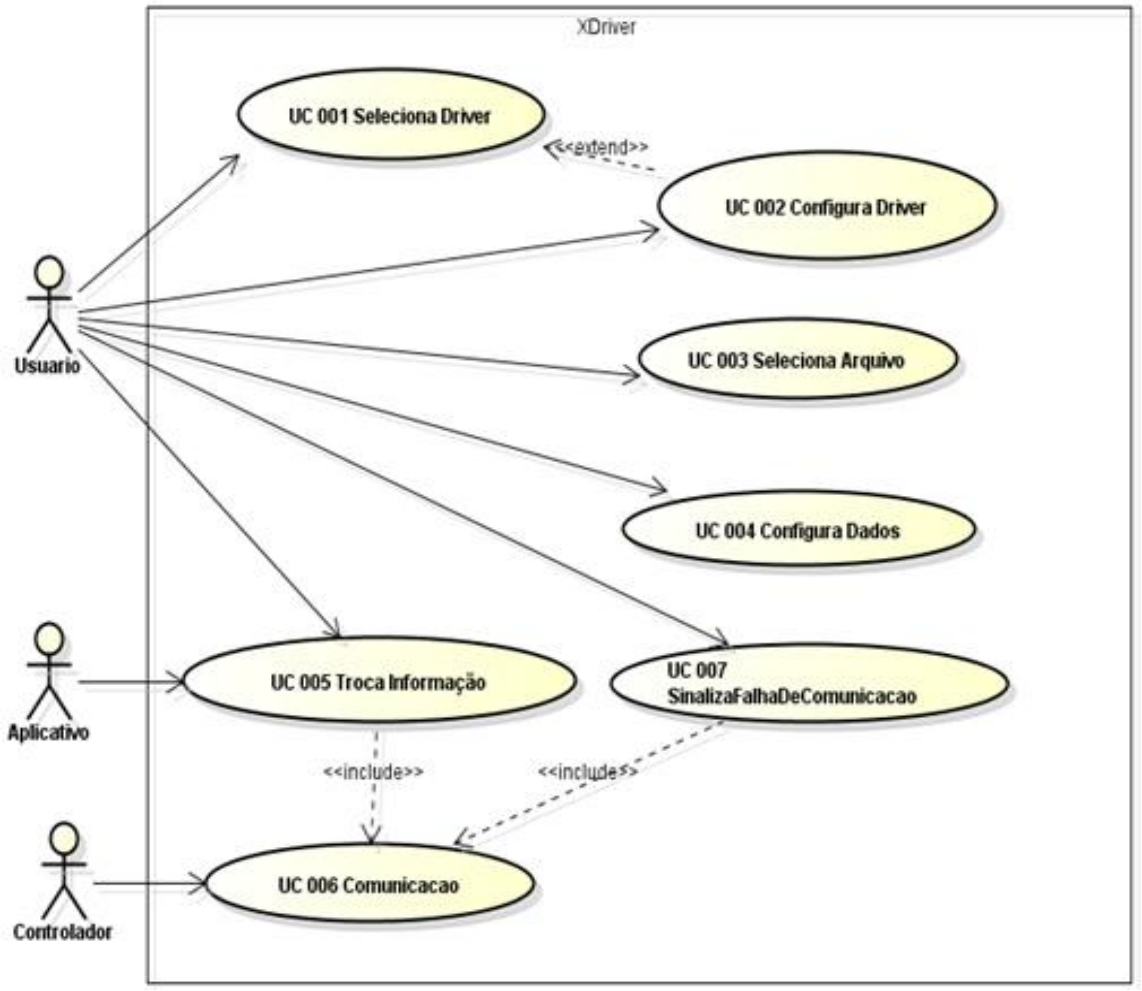

FIGURA 13: Diagrama de caso de uso

O projeto desenvovido, envolve não só o aplicativo principal XDriver, mas também um série de outros sistemas de *hardware* e *software* que serão necessários para a fase de testes da ferramenta durante o seu desenvolvimento, o projeto divide-se em objetos distintos, com funções diferentes entre si, mas que juntos, iram se integrar e interagir na solução do problema proposto.

As etapas ou módulos de desenvolvimento em paralelo foram os seguintes: i) Desenvolvimento do driver de comunicação ModbusIP, DLL com protocolo implementado dentro de padrões que devem ser seguidos pelos futuros protocolos a serem desenvolvidos, afim de possibilitar o intercâmbio de *Drivers* na interface de configuração da ferramenta XDriver; ii) Desenvolvimento das classes do aplicativo XDriver, aplicativo utilizado para troca de dados, o obejetivo principal do projeto, estas por sua vez foram desenvolvidas de forma paralela e posteriormente foram integradas e testadas entre si; iii) Desenvolvimento de rotinas VBA em

Planilha Excel, para demosntrar e exemplificar as funcionalidades do aplicativo desenvolvido; iv) Elaboração, desenvovimento e configuração de rotina de instruções em um CLP com suporte a ModbusIP, para os testes do aplicativo desenvolvido.

Para o desenvolvimento do XDriver, foram definidos os seguintes requisitos funcionais (RF) a serem implementados:

RF001:

O sistema deve prever a seleção e troca de *driver* de comunicação mantendo as demais configurações de arquivo e dados inalteradas quando desejado.

RF002:

O driver deve ter uma interface de configuração acessível ao usuário, esta deve ser acessada pela chamada padrão de interface de *driver* do aplicativo.

RF003:

O sistema deve prever uma seção para o tratamento e configuração de arquivos, com a finalidade de receber as informações dos controladores, CLPs.

RF004:

O sistema deve prever uma seção para o tratamento e configuração dos dados que serão trocados entre aplicativo e controlador.

RF005:

O aplicativo deve enviar dados do controlador para o arquivo selecionado e deste para o controlador conforme configurado na seção de dados.

RF006:

O aplicativo deve prever a comunicação com Controladores Lógicos Programáveis diferentes de maneira padronizada, através do uso de *driver* com protocolo compatível ao controlador utilizado.

RF007:

O sistema deve interromper o processo de comunicação e gerar aviso ao usuário durante um erro de comunicação com o controlador.

3.3 Etapa de Aprendizado

De acordo com Pressman (2006) a etapa de aprendisado significa a experiência obtida com a entrega parcial das etapas do produto. Para o mesmo autor são repetidas revisões de qualidade e foco na demonstração das funcionalidades. Nesta etapa o produto encontra-se na sua concepção final, apartir dos resultados obtidos através dos testes, as etapas anteriores serão analizadas e os

arquivos de documentação serão alterados conforme as correções ou atualizações de *software* venham a ocorrer.

O usuário terá a opção de seleção e configuração dos *drivers* implementados com os protocolos de redes conforme indicado na Figura 14.

FIGURA 14: Tela utilizada para a seleção e configuração de *Drive*

Neste caso o ModbusIP será o *driver* disponível para as aplicações iniciais do projeto, demais protocolos podem ser implementados em DLL seguindo um padrão definido de chamadas para as funções de leitura e escrita de dados. Desta forma implementar protocolos diferentes não implicará na estrutura das demais funcionalidades e configurações já determinadas na área de troca de dados pelo usuário.

Outra função importante disponível é a opção para seleção de planilhas para troca de informação, a planilha que será usada como interface para elaboração de aplicativos SCADA, que segundo Cândido (2004) compreende um meio onde se traduz o que está acontecendo na máquina ou no sistema, um dispositivo que está entre o homem e a máquina. A interface para a seleção de arquivos é mostrada na Figura 15.

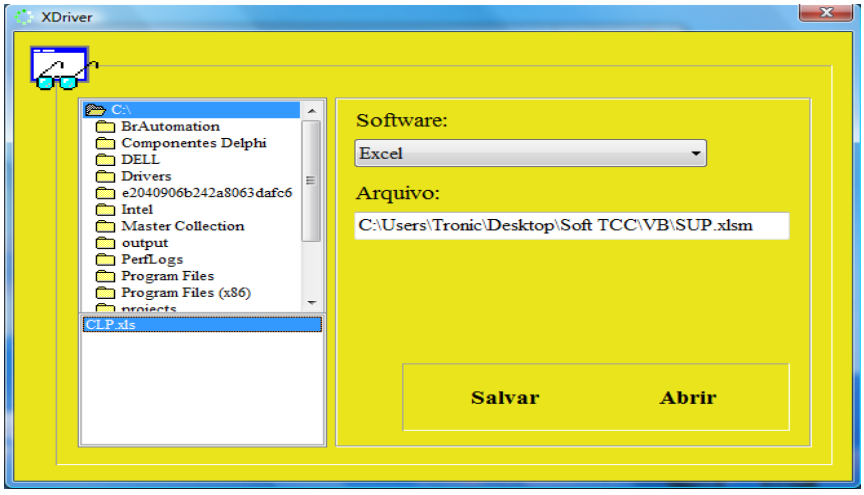

FIGURA 15: Tela utilizada para a seleção de arquivos

Através da seleção de planilhas o usuário irá definir o arquivo destino assim como a configuração das células e tipos de dados a serem trocados entre aplicativos conforme pode ser visto na Figura 16.

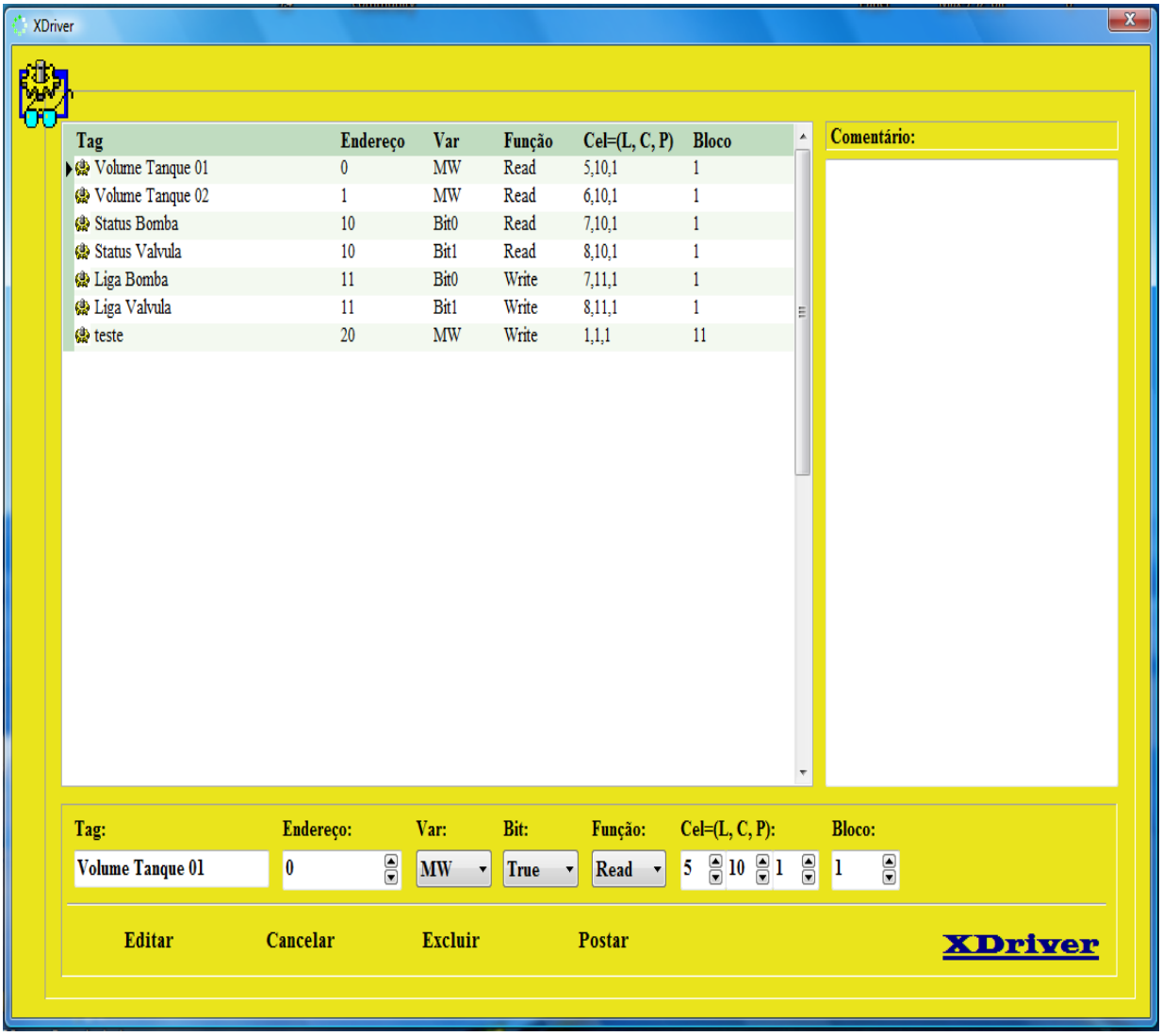

FIGURA 16: Tela utilizada para a configuração de dados

4. Resultados

Como resultado foi obtido o aplicativo denominado XDriver, que conforme o propósito do projeto estabelece a comunicação com dispositivos de controle usados em automação industrial conhecidos como PLCs e *softwares* comerciais que possuem suporte ao padrão COM Microsoft, neste caso uma planilha eletrônica do Excel.

A Figura 17 mostra a tela inicial do aplicativo e seus ícones de acesso as ferramentas de configuração.

FIGURA 17: Interface de serviço do aplicativo XDriver

Basicamente o *software* desenvolvido faz a troca de informação entre aplicativos distintos com funções diferentes, mas que uma vez integrados pela ferramenta, irão atuar em conjunto. De um lado esta o controlador lógico PLC gerenciando uma máquina ou dispositivo qualquer na indústria ou comércio, na outra extremidade uma aplicação do Excel atuando como uma interface para troca de dados e visualização das variáveis de processo do sistema.

Na tela principal da ferramenta esta disponível o acesso as funcionalidades do *software*, assim como a visualização do status de comunicação com o dispositivo a ser monitorado, em caso de falhas durante a transferência de dados entre o XDriver e o CLP, ocorrera uma falha sinalizada pela tela representada na Figura 18.

FIGURA 18: Mensagem de falha na comunicação

A Figura 19 mostra a implementação em uma planilha eletrônica do Excel utilizada como protótipo de teste da ferramenta desenvolvida.

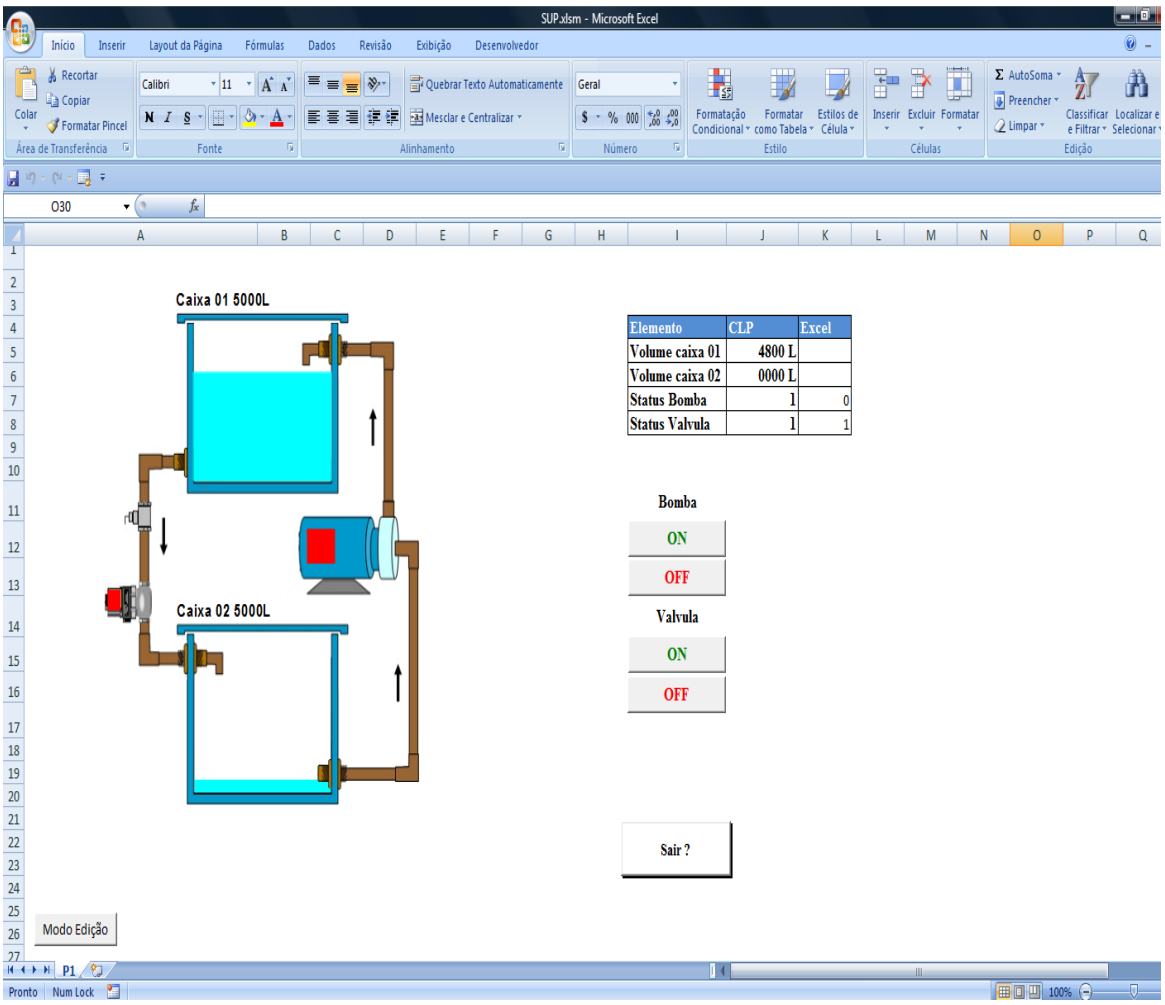

FIGURA 19: Aplicativo Excel

O objetivo é representar um circuito hidráulico com status de uma válvula solenóide para dreno do reservatório superior (caixa 01) e uma bomba elétrica que retira o fluido do reservatório inferior (caixa 02), no lado direito superior da tela fica localizada uma tabela que mostra os valores das variáveis de processo, assim como, o status dos dispositivos e nível dos reservatórios.

Esta aplicação utilizando a panilha do Excel como base para desenvovimento e a interface COM para troca de informação com XDriver se comportou de forma estável mostrando os valores das variáveis em tempo real, conforme o esperado na fase de projeto.

Da mesma maneira, do outro lado no canal de comunicação com o controlador CLP, utilizando o protocolo MdbusIP, a comunicação se manteve estável ocorrendo erros de comunicação somente em simulação de falhas no meio físico da rede, como falta de energia no controlador e rompimento do cabo de comunicação.

Conforme visto anteriormente, a integração entre os aplicativos se da por duas vias distintas, de um lado o padrão COM entre Excel e Xdriver, na outra via o protocolo ModbusIP entre XDriver e CLP, este selecionavel na ferramenta pois como afirmado nos comentários de Martins (2010)

fabricantes diferentes de equipamentos geralmente usam protocolos diferentes para a comunicação entre seus dispositivos.

As arquiteturas e padrões em redes industriais destinam-se a integrar os componentes de um sistema de automação e são caracterizadas em função do protocolo utilizado (MARTINS, 2010).

A Figura 20 ilustra de forma mais clara as vias distintas de comunicação do aplicativo desenvolvido e a ideia de flexibilidade de protocolos de comunicação com CLPs, sem alterações nas demais partes de *software* envolvidas na configuração do XDRiver.

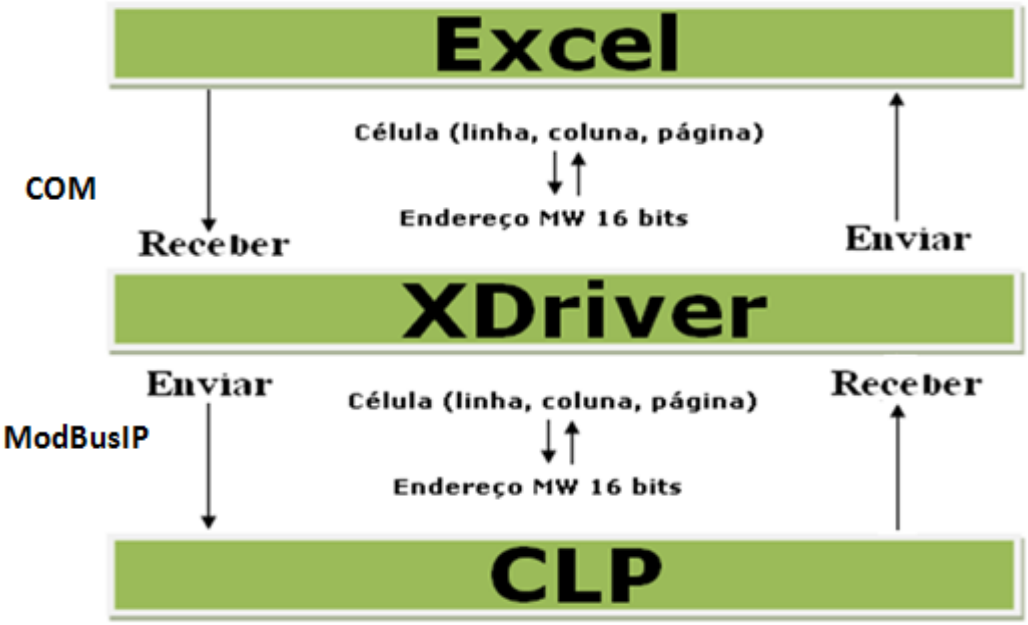

FIGURA 20: Vias de comunicação utilizadas pelo aplicativo XDriver

A Figura 21 mostra a estrutura que representa os componentes do sistema envolvidos no desenvolvimento do projeto, estes por sua vez divididos em módulos diferentes de acordo com suas funcionalidades.

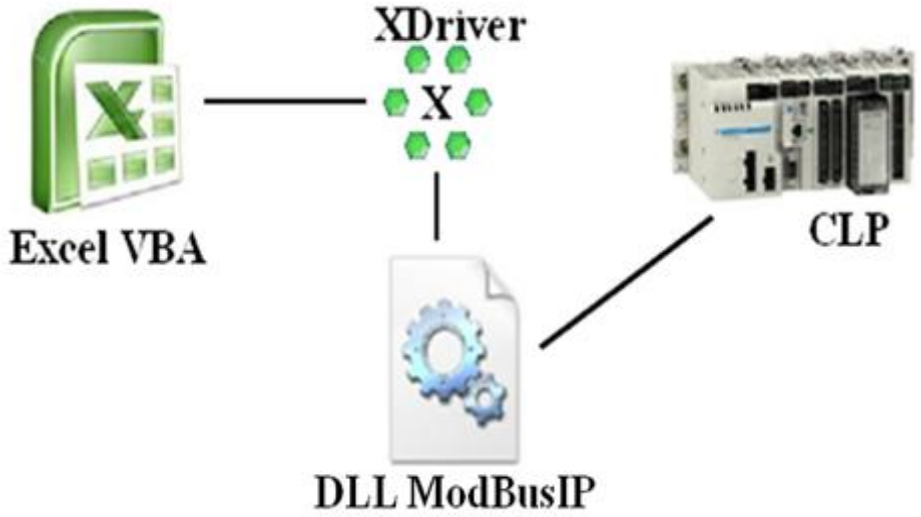

FIGURA 21: Componentes que integram o sistema

5. Conclusão

O aplicativo desenvolvido permite o ajuste flexível dos protocolos de comunicação com CLPs, através da substituição e configuração de *Drivers*, DLLs, que neste caso representam as partes de *software* com funções de acesso a dados em PLCs. Estas partes por sua vez, são as funções que darão o suporte necessário aos protocolos de comunicação utilizados em equipamentos eletrônicos no controle e aquisição de dados.

O resultado foi a criação de um gerenciador que administra a troca de dados entre a aplicação comercial e o controlador industrial em tempo real, tratando exceções e gerando mensagens de erros quando eventuais falhas de comunicação venham a ocorrer, sejam elas causadas por agentes externos, como falta de energia no equipamento monitorado ou eventuais erros durante a troca de dados.

Segundo Tanenbaum (2010) a comunicação de dados em tempo real se refere a sistemas em que o tempo de execução é constante, ou seja, quando um determinado período de tempo é estipulado pelo usuário, se a ação deve acontecer absolutamente em um certo momento ou em um certo intervalo, tem-se um sistema em tempo-real. A ferramenta corresponde a esta requisição de forma estável, dentro do período estipulado, fato que tem ocorrido durante os testes do aplicativo XDriver.

O objetivo de criar um aplicativo de configuração fácil, amigável e intuitivo ao usuário, foi atingido. Desenvolveu-se uma ferramenta de interface simples, permitindo que este possa se dedicar mais a parte visual do aplicativo SCADA, sem se incomodar com o gerenciamento de protocolos de comunicação e aquisição de dados em dispositivos de *hardware*.

A camada de comunicação entre o aplicativo desenvolvido e o aplicativo comercial que irá interagir com o usuário foi padronizada conforme a proposta do projeto. Caso ocorra uma alteração ou troca de protocolo de rede, não sera necessário fazer alterações na camada de endereçamento de dados compartilhados, uma simples troca e configuração do novo *driver*, será o suficiente para mudar o protocolo de rede e interagir com o novo equipamento.

Tornou-se possível através do aplicativo XDriver, que recursos visuais e funcionais de alto nível, existentes em *softwares* como o Office Excel pudessem ser direcionados para fins de automação na área da indústria, sem a necessidade de grandes configurações por parte do usuário.

Ao final foi obtido um sistema capaz de atuar em segundo plano no sistema operacional, despercebido do usuário final, um *software* executável local que interage com controladores industriais, escrevendo e lendo valores na área de memória destes equipamentos.

Referências

BALLOCK, Ivan Roberto. **PROTÓTIPO DE UM SISTEMA PARA SUPERVISÃO DE EQUIPAMENTOS INDUSTRIAIS.** Trabalho de conclusão de curso Ciência da Computação**,** Universidade Regional de Blumenau, Santa Catarina, 2003.

BOLZANI, Caio Augustos Morais. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes**. Tese de Doutorado em Engenharia Elétrica, Escola Politécnica São Paulo, 2010.

CÂNDIDO, Ronei Vilas Bôas. **PADRÃO OPC: Uma Alternativa de Substituição dos Drivers Proprietários para Acessar Dados de PLCs.** Monografia Bacharelado em Ciência da Computação, Universidade FUMEC, Belo Horizonte Minas Gerais, 2004.

DJIEV, Stancho. **Industrial Networks for Communication and Control.** Technical University of Sofia, 2003.
Disponível em: <http://anp.tu-sofia.bg/djiev/Networks.htm>
Acesso em: 14 de março de 2012.

GUIMARÃES, HUGO Casati Ferreira. **Norma IEC 61131-3 Para Programação de Controladores Programáveis: Estudo e Aplicação.** Projeto de Graduação, Universidade Federal do Espírito Santo Vitória, 2005.

KOBAYASHI, Tiago Hiroshi. **Uma Ferramenta de Manipulação de Pacotes para Análise de Protocolos de Redes Industriais Baseados em TCP**/IP. Dissertação de Mestrado em Engenharia da Computação, Universidade Federal do Rio Grande do Norte, 2009.

MARTINS, Marco Roberto Alves. **Integração de Sistemas em Middleware**.
Dissertação de Mestrado em Engenharia Elétrica, Escola Politécnica São Paulo, 2010.

NOGUEIRA, Tiago Augusto. **Redes de Comunicação para Sistemas de Automação Industrial.** Monografia Engenharia e Controle de Automação, Universidade Federal de Ouro Preto, Minas Gerais, 2009.

OLIVEIRA, Daniel. Barbosa. **Programa de Computador para Análise de Relações entre Estrutura Química e Atividade Biológica.** Dissertação de Mestrado em Física, Universidade Federal do Espírito Santo, 2001.

PRESSMAN, Roger. **Engenharia de Software**. 6ª ed. Rio de Janeiro: McGrawHill, 2006.

ROCKWELL. **Tendências da Automação Industrial**. Disponível em:
<http://www.cefetrn.br/~walmy/Rockwell.pdf> Acesso em: 12 de março de 2012.

SILVA, Junior. **Microcontrolador 8051 (Hardware e Software).** 3° ed. São Paulo: Editora Érica, 1994.

SILVA, João Vitor. **Sistemas Administrativos Integrados.** Curso de Ciência da Computação da Faculdade de Jaguariúna, São Paulo, 2007.

SILVA, Ana Paula Gonçalves; SALVADOR, Marcelo. **O que são sistemas supervisórios?** - 2004.
Disponível em: <http://www.wectrus.com.br/artigos/sist_superv.pdf>
Acessado em: 10 de março de 2012.

SILVEIRA, Paulo Rogério; SANTOS, Winderson Eugenio. **Automação e controle discreto.** 5º ed. São Paulo: Editora Érica, 2003.

TANENBAUM, Andrew Stuart. **Sistemas Operacionais Modernos.** 3° ed. São Paulo: Person Education do Brasil, 2010.

TIBOLA, Leandro RosniaK. **Geração de Sistemas Supervisórios a partir de Modelos Orientados a Objetos.** Dissertação de Mestrado em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.