CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

EDUARDO DULLIUS

SISTEMA DE MONITORAÇÃO E IDENTIFICAÇÃO DE DADOS PARA REDES INDUSTRIAIS PADRÃO DEVICENET

Lajeado
2008

EDUARDO DULLIUS

SISTEMA DE MONITORAÇÃO E IDENTIFICAÇÃO DE DADOS PARA REDES INDUSTRIAIS PADRÃO DEVICENET

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia de Controle e Automação.
Área de concentração: Automação Industrial

ORIENTADOR: Ronaldo Hüsemann

Lajeado

2008

EDUARDO DULLIUS

SISTEMA DE MONITORAÇÃO E IDENTIFICAÇÃO DE DADOS PARA REDES INDUSTRIAIS PADRÃO DEVICENET

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia de Controle e Automação pelo CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Ronaldo Hüsemann, UNIVATES

Mestre pela UFRGS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Robson D. Schaeffer, UNIVATES

Mestre pela UFRGS – Porto Alegre, Brasil

Prof. Vinicius L. Curcio, UNIVATES

Mestre pela UFSC – Florianópolis, Brasil

Coordenador do curso de Engenharia de Controle e Automação :_____

Prof. Robson D. Schaeffer

Lajeado, junho de 2008.

Dedico este trabalho a meus pais, Waldemar e Mercedes, pelo incondicional apoio em momentos difíceis, e a meu irmão Rodrigo por sua consideração.

AGRADECIMENTOS

Ao meu orientador Ronaldo Hüsemann, por oportunizar a realização de trabalhos em minha área de pesquisa.

Agradeço também aos colegas e amigos Gustavo Gasparini, Gustavo Künzel e Henrique Tiggemann por auxiliarem nas atividades desenvolvidas neste período.

Também aos professores e demais colegas do curso que de alguma forma me apoiaram e auxiliaram ao longo do desenvolvimento deste trabalho.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de monitoração e identificação de dados para redes industriais padrão DeviceNet. O texto inicia apresentando a importância de redes de comunicação em ambientes industriais, enfocando as características do protocolo CAN e mais especificamente o padrão de rede DeviceNet. Tendo em vista a importância da área de validação de redes industriais, propõe-se o desenvolvimento de um sistema de monitoração e identificação de mensagens DeviceNet. Sua especificação foi orientada em outros trabalhos acadêmicos bem como soluções em monitoração e análise disponíveis atualmente no mercado. Por fim, apresentam-se diversos ensaios práticos que comprovam o seu funcionamento em condições reais de operação.

Palavras-chaves: Redes Industriais, Protocolo CAN, DeviceNet.

ABSTRACT

This work presents the development of a data monitoring and identifying system for industrial networks based on DeviceNet standards. The document first shows the communication networks importance in industrial environments, focusing the fundamentals of CAN protocol and DeviceNet standards. Considering the area importance, it is proposed the development of a DeviceNet message monitoring and identifying system. The system specification was guided by academic works and some solutions for monitoring and analysis currently available in the market. Finally some practical experiments are presented validating the proposed system to work in real operation conditions.

Keywords: Industrial networks. CAN protocol. DeviceNet.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACK: Acknowledgement

A/D: Analógico/Digital

CAN: Controller Area Network

CiA: CAN in Automation

CIP: Common Industrial Protocol

CLP: Controlador Lógico Programável

COS: Change of State

CPU: Central Process Unit

CRC: Cyclic Redundancy Code

DAS: Device Access Server

DLC: Data Length Code

EDS: Electronic Data Sheet

EIA: Electronic Industries Alliance

EOF: End Of Frame

IHM: Interface Homem-Máquina

ID: Identificador

IDE: Identifier Extension Bit

IEC: International Electrotechnical Commision

IEEE: Institute of Electrical and Electronics Engineers

ISO: International Organization for Standardization

I/O: Input/Output

MAC: Media Access Control

NRZ: Non Return-to-Zero

ODVA: Open DeviceNet Vendor Association Inc.

OSI: Open Systems Interconnection

PCI: Peripheral Component Interconnect

RAM: Random Access Memory

RTR: Remote Transmition Request

SAE: Society of Automotive Engineers

SOF: Start Of Frame

SRR: Substitute Remote Request

TTL: Transistor–Transistor Logic

UCMM: Unconnected Message Manager

USB: Universal Serial Bus

1 INTRODUÇÃO

Tendo em vista um mercado cada vez mais competitivo, onde as dificuldades impostas pela concorrência e as exigências dos consumidores aumentam diariamente, as indústrias não podem perder tempo. A idéia de uma empresa ser concebida pensando em produzir determinado produto e assim continuar durante toda sua existência não é mais aceitável. O mercado, como um todo, funciona impulsionado por inovação. Padrões de exigência da sociedade levaram ao conceito de que itens manufaturados devem ser modificados e renovados periodicamente e com grande freqüência, tornando cada vez mais curto o ciclo de vida de produtos (ROSÁRIO, 2005).

O que surge então é a necessidade de linhas de produção flexíveis e que atendam a eventuais mudanças de uma maneira prática e com custo reduzido. Entretanto, o tempo disponível para adaptações ou aprimoramentos se mostra cada vez menor. Em paralelo a isso, a fim de aumentar o controle de qualidade na produção, cada vez mais exige-se a integração de sistemas do ambiente industrial com os de gerência. Assim, os dados referentes à produção devem trafegar desde o chão de fábrica até os níveis mais altos da hierarquia gerencial. Nesse contexto, abre-se um enorme campo de atuação na área de automação.

Em sistemas de automação tradicionais os equipamentos no chão de fábrica transferem dados através de cabos exclusivos para cada sinal. Com isso cada vez que um novo sinal é adicionado ao sistema, um novo par de fios deve ser instalado entre o dispositivo sensor e o controlador ou a placa de interface. Essa forma convencional de cabeamento, porém, apresenta uma série de problemas, que se acentuam na medida em que a complexidade do sistema aumenta. Dentre as principais desvantagens pode-se citar a dificuldade para localização de defeitos nos cabos e dificuldade para expansão ou mesmo alteração do sistema quando em funcionamento. Com isso surgem dificuldades na avaliação de desempenho e diagnóstico do sistema. Dependendo da distância, o valor do cabeamento também se mostra bastante elevado, assim como sua manutenção.

Com o uso da tecnologia de redes de comunicação industrial, os problemas citados são reduzidos, dando lugar a um cabeamento mais simples, econômico e confiável. As instalações nas linhas de produção se tornam mais flexíveis, possibilitando agilidade e segurança em caso de eventuais mudanças. Os dispositivos de campo deixam de ser apenas coletores de dados para também atuarem no tratamento desses dados. O envio dos respectivos valores é feito em forma de pacotes de mensagens, uma vez que os equipamentos agregam agora inteligência própria para controle e comunicação. O sistema de gestão também sai favorecido, pois novas

funcionalidades por parte do planejamento e controle da produção se tornam viáveis (ERIKSSON; COESTER; HENNIG, 2006).

Levando em conta esses aspectos de flexibilidade, agilidade e confiança desejados para linhas de produção, o mercado cada vez mais vem substituindo a automação convencional, por aplicações de automação integradas a redes de comunicação industriais. Sendo assim, sistemas de comunicação de dados vêm se tornando um componente essencial em plantas de automação, implantando um conceito inovador que está crescendo constantemente no campo industrial (NATALE, 2000).

Mas a disseminação do uso de redes de comunicação em ambientes industriais também ocasionou o surgimento de um grande número de protocolos diferentes, provenientes de diversos fabricantes e associações. Algumas redes são definidas para uso genérico enquanto outras têm aplicações bem específicas.

Há redes indicadas para conexão somente entre Controladores Lógicos Programáveis (CLP) e dispositivos de campo, como sensores, atuadores, transmissores, *drivers*, Interfaces Homem-Máquina (IHM), etc. Outras já são mais adequadas para comunicação entre CLPs e computadores. E há também as redes cujo uso não se restringe unicamente ao nível de campo (planta de processos), sendo indicadas para o envio de dados aos níveis de gerenciamento da organização (setor administrativo), permitindo que decisões sejam tomadas com base em dados confiáveis e constantemente atualizados. As formas de implementação e demais características como taxa de transferência de dados, comportamento temporal, número máximo de nós e comprimento do barramento, variam de rede para rede. Sendo assim, é preciso conhecer o processo onde a rede será usada, juntamente com suas especificações, para só então optar pelo mecanismo mais apropriado.

Dentre os protocolos de comunicação disponíveis comercialmente na atualidade, um de grande destaque é o protocolo CAN (*Controller Area Network*). Surgido nos anos 80 como solução para aplicações automotivas, em poucos anos se tornou extremamente difundido no meio industrial, proporcionando as mais variadas aplicações (KIRSCHBAUM et. al, 1996).

As normas que definem o CAN não determinam todo o seu modo de funcionamento. O CAN especifica parte das camadas física e de enlace, deixando que desenvolvedores de sistemas de mais alto nível especifiquem o restante dessas camadas e também a camada de aplicação. Citando os mais difundidos, é o caso dos protocolos CANopen, CANKingdom e DeviceNet, sendo esse último o foco do presente trabalho.

A importância do protocolo CAN pode ser percebida observando-se uma pesquisa feita pelo IMS Research, divulgada em janeiro de 2007. O IMS é um dos mais importantes institutos de pesquisa para a indústria eletrônica, publicando em torno de 100 pesquisas anualmente nas áreas de semicondutores, automação industrial, comunicação, eletrônica de potência, segurança, identificação, equipamentos médicos, entre outros. Sua pesquisa, intitulada "*The Market for Industrial Networking 2006 – Worldwide*", apresenta um sólido crescimento em tecnologias de comunicação industrial (IMS, 2007).

São apresentados os totais de nós instalados ao redor do mundo em 2006, separados por protocolo, dando destaque aos mais difundidos, conforme a figura 1. Nota-se aqui a grande quantidade de unidades CAN e DeviceNet instalados no ano em questão (mais de dois milhões somando-se ambas as soluções). Outros dados levantados também pela entidade IMS, indicam um crescimento anual em torno de 13% no mercado de redes industriais durante pelo menos os próximos cinco anos.

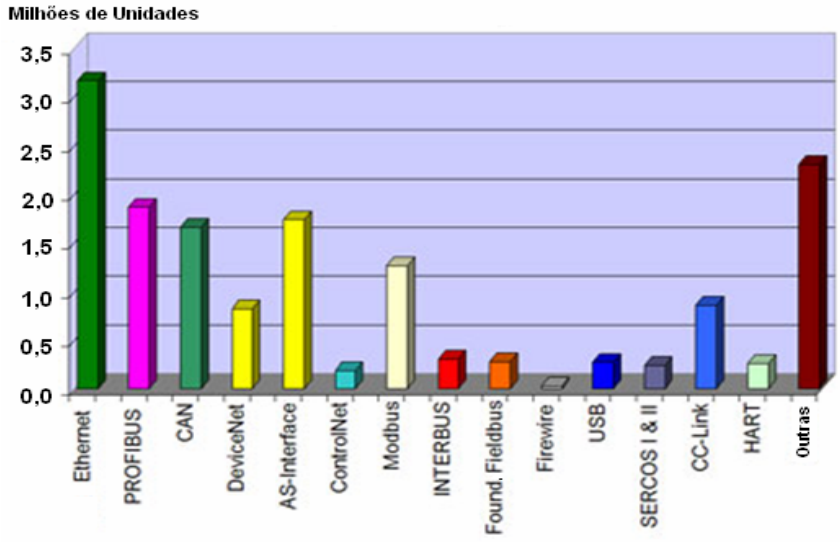


Figura 1 Total de nós por protocolo instalados em 2006.

Com base no contexto apresentado, pode-se perceber claramente a importância das redes em ambientes industriais e sua crescente utilização na área de automação industrial. Esta situação serviu de motivação para o desenvolvimento do sistema aqui proposto. Escolheu-se como tecnologia alvo o barramento DeviceNet, por se tratar de um barramento bastante importante no mercado (conforme a figura anterior) e também pelo fato de haver CLPs com interface para redes DeviceNet disponíveis no Laboratório de Automação Industrial da Univates.

Particularmente apresenta-se neste trabalho a proposta de um sistema de monitoração e identificação de mensagens em barramento DeviceNet. O sistema é composto de duas partes: um dispositivo em hardware para captura de dados e um software de identificação e exibição de mensagens.

O princípio de funcionamento do sistema é bastante simples, a exemplo de outras soluções atualmente disponíveis no mercado (HUSEMANN, 2007). O dispositivo de hardware é conectado à rede de comunicação capturando em tempo de execução as mensagens que passam pela rede. O dispositivo opera de maneira passiva, apenas capturando dados e armazenando em sua memória interna. Os dados adquiridos são transmitidos para um computador através de interface serial ou USB, para serem posteriormente identificados. A ferramenta em software desenvolvida permite a identificação e exibição de diversas informações relevantes tais como grupo da mensagem, identificador MAC, tamanho e conteúdo do campo de dados, significado da mensagem, tempo de ocorrência entre outras.

Com isso o sistema proposto deve auxiliar na compreensão da situação real de operação de rede em condição real de operação, bem como trazer novos subsídios para desenvolvedores de aplicações em DeviceNet a título de validação ou mesmo correção de problemas de configuração.

Resumidamente, este trabalho está dividido da seguinte forma: o capítulo 2 traz uma apresentação das redes industriais de forma genérica; no capítulo 3 aprofunda-se no protocolo CAN, apresentando-se suas características e modo de funcionamento; o capítulo 4 apresenta mais especificamente as características de uma rede DeviceNet; no capítulo 5 são citados trabalhos acadêmicos envolvendo o tema em foco e mostrados produtos disponíveis atualmente no mercado, cujas funcionalidades se assemelham ao que será proposto; no capítulo 6 apresenta-se a descrição do presente trabalho, incluindo as etapas de especificação e implementação do sistema montado; o capítulo 7 mostra ensaios realizados em redes DeviceNet reais, a fim de demonstrar o funcionamento da ferramenta implementada; por fim, no capítulo 8 são apresentadas as conclusões bem como possíveis desdobramentos dos trabalhos aqui apresentados.

2 REDES INDUSTRIAIS

Até a década de 70, o uso de controladores e computadores em ambientes industriais limitava-se basicamente ao chão de fábrica, no nível dos instrumentos de campo, para coleta de dados usados no monitoramento e controle de plantas. A transferência de dados entre o chão de fábrica e a parte gerencial não era possível, a não ser manualmente (ERIKSSON, 2006).

Entretanto, com o crescente aprimoramento do controle de processos e busca por melhores resultados, surgiu a necessidade de uma transferência de dados direta e uma interatividade entre os computadores através de uma interface de comunicação, relacionando os dados coletados em diferentes níveis da automação. Tal implementação, entretanto, enfrentou inicialmente muitas dificuldades. Um dos principais problemas enfrentados era que os vários desenvolvedores de softwares e equipamentos não proviam compatibilidade entre seus produtos (BEGA et. al., 2006).

Estudos desenvolvidos pelos próprios fabricantes levaram ao surgimento de formas de comunicação entre seus produtos, mas continuava o problema da incompatibilidade com outros fabricantes, pois cada fabricante criava uma solução própria, não tornando pública as formas de implementação. A partir deste conceito, diversas interfaces diferentes foram desenvolvidas por companhias de automação para serem lançadas no mercado. Como exemplos desses primeiros barramentos desenvolvidos podem ser citados: o HIGHWAY da Honeywell, DATA HIGHWAY da Fischer Controls, FOXNET da Foxboro, FEEDER BUS da BBC (PARK; MACKAY; WRIGHT, 2003).

Estas soluções aumentavam a capacidade de controle, porém ainda mantinham os clientes amarrados ao fabricante. Além disso, dificilmente um único desenvolvedor disponibilizava equipamentos para atender da melhor forma uma planta de automação por completo. Por esse motivo, hoje os protocolos proprietários estão em desuso.

Nos anos 80, graças a um longo trabalho do *Institute of Electrical and Electronics Engineers* (IEEE) e da *International Organization for Standardization* (ISO), os progressos na padronização de redes de comunicação trouxeram ao mercado algumas formas de transferência de dados abertas, permitindo a implementação de controles distribuídos em plantas de automação hierarquicamente organizados. Padronizações criadas pelo IEEE na área de redes locais e pela ISO com o Modelo OSI (*Open Systems Interconnection Model*), determinaram os fundamentos de novas tecnologias de comunicação de dados (POPOVIC; VLACIC, 1999).

O modelo OSI não é um protocolo ou um conjunto de regras ditando como um protocolo deve ser escrito. O modelo é essencialmente uma estrutura gerencial, que promove uma base comum para o desenvolvimento de padrões, especificando a comunicação de dados dentro de uma hierarquia de sete camadas (níveis), conforme a figura 2.

Figura 2 As sete camadas do modelo OSI

Este modelo serviu de base para a construção de sistemas abertos, os quais, ao contrário dos sistemas fechados, apresentam diretrizes e especificações publicamente disponíveis para implementação por aqueles que desejarem, permitindo assim que dispositivos de vários fabricantes diferentes possam trabalhar em conjunto em um mesmo ambiente (TANEMBAUM, 2003).

Cada camada tem um propósito definido, que depende das camadas que estão acima e abaixo dela, porém com flexibilidade para sua própria implementação, permitindo que desenvolvedores trabalhem em cada camada do sistema de forma independente (SOARES; LEMOS; COLCHER, 1995).

Nível 1 - Físico: Na camada física são definidos aspectos elétricos, tais como tensão e corrente do sistema e também aspectos mecânicos das conexões, como cabos e conectores. Sendo que nesse nível não são tratados erros de transmissão envolvendo sua detecção ou correção, sua função é proporcionar o envio de uma seqüência de bits pela rede, sem se preocupar com o seu conteúdo ou de que forma esses bits serão agrupados.

Nível 2 - Enlace: Um dos principais objetivos do nível de enlace é a detecção de erros ocorridos durante a transmissão de dados através do nível físico. Para tanto, realiza a divisão da cadeia de bits em pacotes, cada um contendo alguma forma de redundância para detecção de erros, normalmente com a adição de alguns bits. Também é no nível de enlace que são

realizados os procedimentos para controle do fluxo de dados entre usuários, realizando a apropriada estratégia de controle de acesso ao meio.

Nível 3 - Rede: É no nível de rede que ocorre o roteamento de mensagens de uma rede para outra. Também é função desse nível definir como os pacotes longos de dados recebidos da camada de transporte são fragmentados em pacotes menores.

Nível 4 - Transporte: Como o nível de rede não garante que todos os pacotes de dados da origem cheguem ao destino e nem mesmo em sua seqüência original, o nível de transporte é quem divide e reagrupa a informação binária em pacotes entre a origem e o destino da informação.

Nível 5 - Sessão: É o nível responsável pela organização e sincronização da troca de dados, permitindo que duas aplicações em estações diferentes estabeleçam um canal de comunicação personalizado.

Nível 6 - Apresentação: É o responsável por converter os dados recebidos pela camada de aplicação em um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado. Esta camada pode implementar mecanismos de codificação ou compressão dos dados.

Nível 7 - Aplicação: É o nível mais alto do modelo OSI, sendo responsável por permitir acesso à rede por parte das aplicações ou interface com usuários. Exemplos das tarefas desse nível incluem transferência de dados e gerenciamento da rede.

Com a intenção de não perder espaço, fabricantes iniciaram então o desenvolvimento de protocolos abertos. Da mesma forma, muitos foram lançados no mercado. Entidades tentaram definir qual seria a melhor opção, especificando assim um único protocolo para aplicações industriais, mas não chegaram a um consenso.

Sendo assim, atualmente estão disponíveis no mercado diversos protocolos diferentes. Entre os mais difundidos pode-se citar: AS-I, CAN, CC-Link, ControlNet, Ethernet, Firewire, Foundation Fieldbus, HART, Interbus-S, Modbus, P-Net, Profibus DP/PA, SERCOS, SwiftNet e WorldFIP. Cada um apresenta características próprias, como comportamento temporal, taxa de transmissão, comprimento de barramento e mecanismo de controle de acesso ao meio, definidas por normas e padrões de entidades internacionais.

3 PROTOCOLO CAN

O procolo CAN (*Controller Area Network*) é um padrão de comunicação definido em normas internacionais, cujas especificações tratam basicamente da camada física do protocolo (nível 1 do modelo OSI) e parte da camada de enlace (nível 2 do modelo OSI). Especificações referentes ao restante da camada física e da camada de enlace, assim como da camada de aplicação (nível 7 do modelo OSI) são deixadas aos desenvolvedores de sistemas de mais alto nível, como as redes DeviceNet e CANopen, entre outras.

3.1 Origem

No início dos anos 80, com o crescente uso de equipamentos eletrônicos em automóveis, a quantidade de cabos existentes no seu interior chegava ao comprimento de alguns quilômetros, tornando os projetos inviáveis. Com isso, engenheiros da empresa alemã Robert Bosch GmbH iniciaram trabalhos para especificação de um protocolo de comunicação serial que possibilitasse troca de dados entre dispositivos embarcados em carros de passeio (CiA, 2007a).

Já no início desta especificação, engenheiros da Mercedes-Benz também se envolveram, ajudando a conceber a idéia. Dr. Wolfhard Lawrenz, professor da Universidade de Ciências Aplicadas de Braunschweig-Wolfenbüttel, na Alemanha, consultor do projeto, foi quem batizou o protocolo de *Controller Area Network*.

Seu lançamento oficial foi em fevereiro de 1986, em um congresso em Detroit. Já na metade de 1987 a Intel entregou o primeiro controlador CAN. Pouco tempo depois a Philips também lançou seu controlador CAN, sendo seguida por outras empresas.

Devido ao sucesso de implantação do CAN no meio automotivo, já na década de 90 iniciaram-se as tentativas de implementação do protocolo em ambiente industrial. Por apresentar diversas características favoráveis, se tornou um padrão utilizado como base para sistemas de comunicação em barramento em vários segmentos de automação.

3.2 Meio Físico

Os parâmetros elétricos do protocolo CAN são definidos pelas normas ISO 11898 (partes 2 e 3), SAE J2411 e ISO 11992. Estes parâmetros são bastante importantes principalmente para barramentos com grande comprimento e velocidade relativamente alta, onde questões como o tempo de propagação do sinal e impedância do cabo são fatores

determinantes. A queda de tensão medida ao longo do cabo deve ser tal que o nível do sinal gerado na origem do sinal seja ainda corretamente interpretado por todos os nós receptores. Existem propostas de transmissão de sinais CAN sobre o canal da alimentação, mas ainda assim a implementação convencional utiliza dois pares de fios separados (dois de dados e dois de alimentação) (CiA, 2007b).

O tempo de propagação do sinal no barramento CAN é determinado pelas duas estações (nós) mais distantes no barramento, sendo definido como o tempo que um sinal leva para viajar de um nó emissor até o nó receptor mais distante e o sinal retornar até o nó emissor. Caso seja preciso estender o tamanho do barramento além do suportado ou aumentar o número de estações presentes, um repetidor pode ser usado. O repetidor, como o próprio nome diz, é um equipamento responsável por ler e transferir bidirecionalmente os sinais elétricos de um segmento do barramento para outro (o sinal é simplesmente repetido).

Outro aspecto relevante a ser observado em uma implementação em barramento, é que os sinais elétricos podem ser refletidos quando chegam ao final do cabo, produzindo interferência no sinal de comunicação. Para se evitar esta interferência destrutiva deve-se utilizar resistores de terminação em ambas as pontas do barramento, cuidado este tomado também por outros protocolos.

Dentre as normas publicadas especificamente para a camada física em redes CAN, a mais difundida é a ISO 11898 parte 2 (*high speed*). Esta norma prevê uma taxa de transmissão máxima de até 1 Mbit/s para um comprimento menor que 40 metros. A comunicação se dá a partir de um barramento diferencial de dois fios, com dois níveis de sinal lógico: CAN_L (*Low level*) e CAN_H (*High level*). A impedância característica do cabo é de 120 Ohms e a voltagem é de -2 V no CAN_L e +7 V no CAN_H.

Já a norma ISO 11898 parte 3 (*fault-tolerant*) é uma variante do protocolo CAN voltada para aplicações onde o nível de robustez do meio físico é mais crítico (indústria automobilística), empregando para tanto uma forma alternativa de interface ao barramento e arranjo de cabeamento.

3.3 Enlace de Dados

Conforme já comentado anteriormente, a forma como as estações estão conectadas no protocolo CAN é através de uma rede em topologia de barramento. Com isso, qualquer mensagem postada por uma dada estação é simultaneamente percebida para todas as outras, pois o meio físico (um único cabo) é compartilhado, conforme a figura 3.

Figura 3 Estação 2 escrevendo uma mensagem no barramento.

Para evitar colisão de mensagens no barramento, o protocolo CAN emprega uma representação de valores composta por bits dominantes e recessivos. Cada pacote de mensagem contém um identificador, que precede a mensagem indicando logo de início o seu conteúdo e prioridade. Por questões de controle de acesso ao meio físico compartilhado, cada identificador é único dentro da rede. As estações ouvintes consomem a informação somente caso seja necessário, interrompendo sua leitura ou não assim que identificam sua necessidade. No exemplo da figura 4, as estações 1 e 4 consomem a informação, enquanto que a estação 3 não consome.

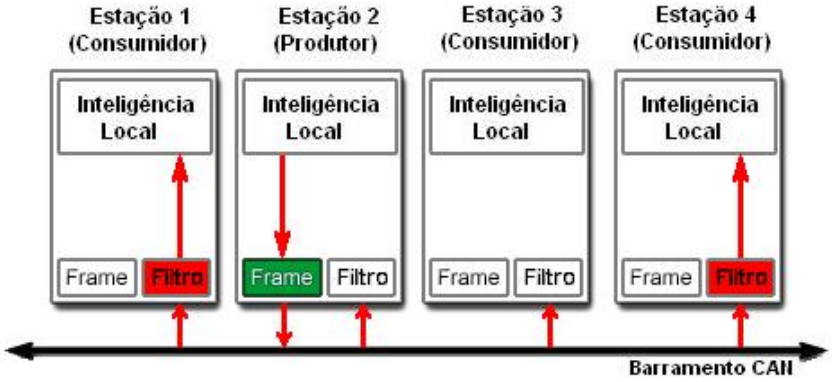


Figura 4 Estações 1 e 4 consumindo a informação.

De forma prática, a compatibilização de diferentes aplicações industriais em uma mesma rede no protocolo CAN se dá a partir da especificação adequada de seus identificadores. Ou seja, o identificador de cada mensagem CAN define, além do seu conteúdo, a prioridade com a qual é feita a transmissão de uma mensagem CAN em comparação a outra menos urgente. As prioridades devem ser definidas a priori pelo projetista

da rede durante o desenvolvimento do sistema, gerando uma relação entre tipos de dados/comando e valores binários correspondentes.

Este conceito de prioridade de mensagens está diretamente ligado com o mecanismo de contenção de colisão da transmissão CAN, composto por bits recessivos (nível alto) e dominantes (nível baixo). Neste contexto o identificador com o menor valor binário apresenta maior prioridade que identificadores que apresentarem um valor binário maior.

O mecanismo CAN de controle de acesso ao meio foi projetado a fim de evitar a colisão e perda de dados no barramento. O protocolo CAN utiliza um método chamado *non-destructive bit-wise arbitration*, onde as mensagens que entram em colisão não são destruídas. Os conflitos de acesso ao meio são resolvidos pelo estado dos bits dos identificadores de cada estação transmissora.

O estado dominante (nível lógico 0) possui privilégio de acesso ao meio sobre o estado recessivo (nível lógico 1). Por isso, todas as estações cujo identificador apresentar um valor recessivo em comparação ao identificador de outra estação concorrente, perdem a competição. Todas as estações perdedoras (com bit recessivo) automaticamente param suas transmissões se tornando receptores da mensagem com a prioridade mais alta e não tentam retransmitir até que o barramento esteja disponível novamente. Por este mecanismo não existe perda de pacotes com colisões, apenas atrasos das mensagens de menor prioridade.

Nas figuras seguintes, está presente um exemplo onde três estações competem pelo acesso ao barramento. Neste caso, está sendo usada uma mensagem com identificador de 11 bits. Na parte superior do desenho estão expostos os bits pertencentes à parte do identificador de cada mensagem. A comparação inicia com o bit mais significativo (bit 10) indo até o menos significativo (bit 0). Percebe-se que até o bit 6, todas as estações apresentam o mesmo valor, mas a partir do bit 5 a estação 1 está diferente, como na figura 5.

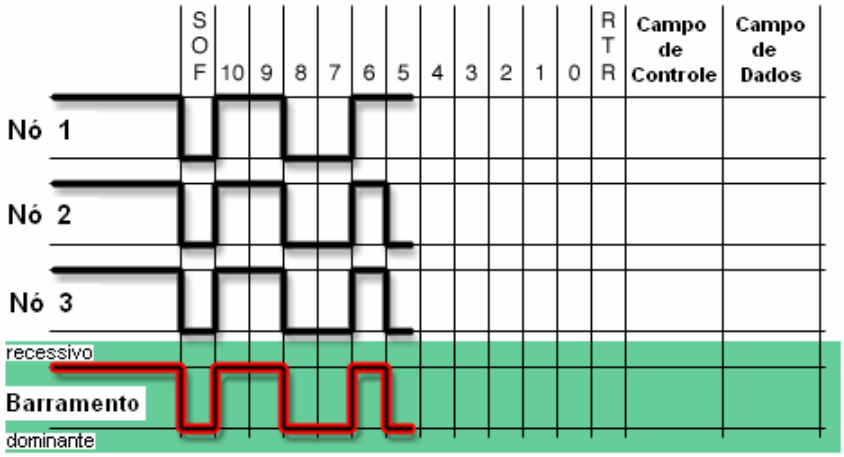


Figura 5 Estações escrevendo no barramento.

Como a estação 1 apresentou um bit recessivo (1) enquanto as outras apresentaram um bit dominante (0), ela abandona a escrita no barramento, e passa a somente ler o que nele trafega. O mesmo acontece com a estação 3, que também apresenta um bit recessivo. Sendo assim, a estação 2, por ter apresentado uma maior prioridade, fica com o acesso ao barramento, visto na figura 6.

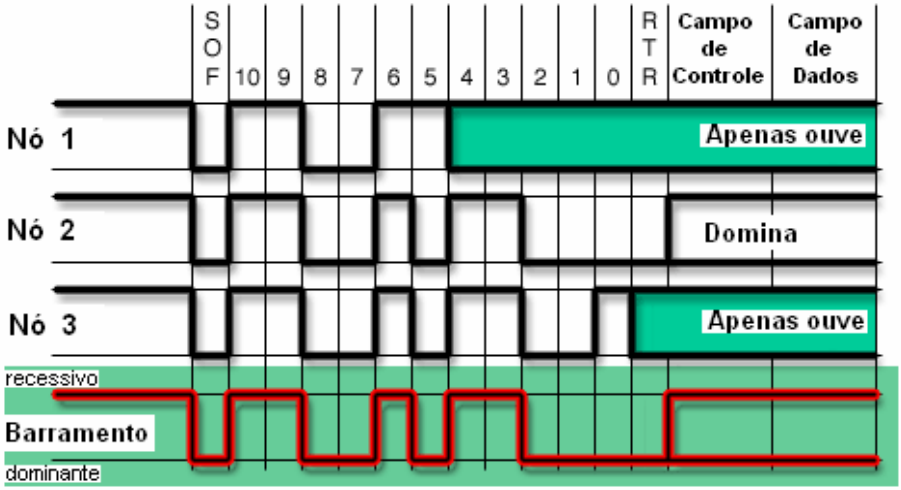


Figura 6 Estação 2 fica com o acesso ao barramento.

Desde que o acesso ao barramento seja definido pela prioridade indicada no identificador da mensagem, é possível garantir um baixo tempo individual de latência em sistemas de tempo real, onde as mensagens com maior importância sempre prevalecerão no acesso ao barramento.

A codificação de bit do protocolo CAN é NRZ (*Non Return-to-Zero*), onde o nível do sinal permanece constante durante o tempo do bit, ou seja, não há alteração do seu valor entre suas bordas de subida e descida. O nível do sinal no barramento pode permanecer constante durante um determinado período de tempo, desde que se garanta que o máximo tempo de intervalo entre duas mudanças de sinal não seja excedido, sendo muito importante para propósitos de sincronização. Para evitar que o sinal fique muito tempo “preso” em um determinado valor, a norma especifica que deve-se inserir um bit complementar depois de cinco bits consecutivos com o mesmo valor.

A transmissão de bits no protocolo CAN ocorre de maneira assíncrona (sem um sinal de relógio adicional). Exige-se assim um método eficiente de sincronização adicional, pois o adequado sincronismo no envio de mensagens é fundamental. Conforme pode-se imaginar, para que o controle de acesso ao meio CAN funcione corretamente, todas as estações concorrentes devem iniciar a transmissão de seus dados ao mesmo tempo, garantindo assim

que a comparação entre suas prioridades ocorra no momento certo. Há dois tipos distintos de sincronização: a sincronização forçada, no início do pacote de mensagem e a ressincronização, durante a transmissão do pacote.

3.4 Pacotes de Mensagens CAN

O protocolo CAN suporta dois formatos de pacotes de mensagens: o formato padrão e o formato estendido. As “mensagens padrão” (CAN 2.0A) prevêem um identificador com tamanho de 11 bits, enquanto as “mensagens estendidas” (CAN 2.0B) usam um identificador com tamanho de 29 bits. Na verdade, os dois formatos iniciam por um identificador de 11 bits, porém o formato estendido acrescenta um segundo identificador de 18 bits, que somado ao identificador inicial totaliza 29 bits. O tráfego de dados no barramento ocorre através de quatro diferentes tipos de mensagens: mensagem de dados, mensagem remota, mensagem de erro e mensagem de sobrecarga (CiA, 2007c).

3.4.1 Mensagem de Dados

As mensagens de dados são usadas para envio dos dados das estações transmissoras para as estações receptoras. São compostas por sete diferentes campos de bits: SOF, campo de arbitragem, campo de controle, campo de dados, campo CRC, campo ACK e EOF. Nas figuras abaixo estão representados os dois tipos de mensagens. A figura 7 mostra a mensagem padrão, com 11bits no identificador, enquanto na figura 8 está representada a mensagem estendida, com 29 bits no identificador.

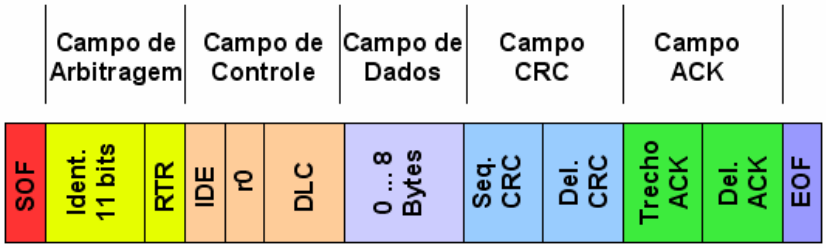


Figura 7 Mensagem de dados padrão.

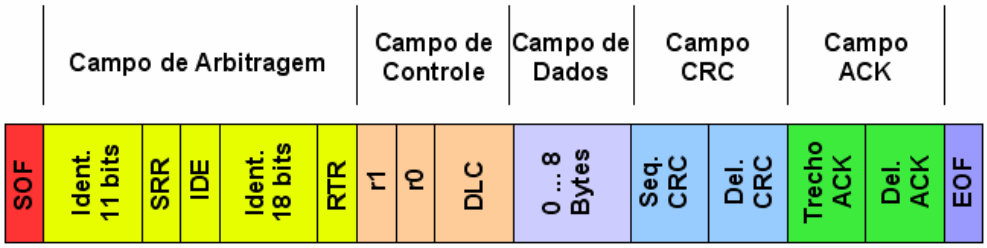


Figura 8 Mensagem de dados estendida.

Abaixo segue uma breve descrição de cada um dos campos de bits que constituem uma mensagem de dados:

SOF (*Start Of Frame*): Marca o início de todas as mensagens, sejam elas mensagens de dados ou mensagens remotas. Consiste em um único bit dominante (nível lógico 0).

Campo de Arbitragem: No formato padrão, é constituído pelos 11 bits no identificador, denominados ID-28 ... ID-18 e mais o bit RTR. Já no formato estendido, é formado por 29 bits dos identificadores (11 e 18), denominados ID-28 ... ID-0 e pelos bits SRR, IDE e RTR.

Identificador: No formato padrão o identificador tem o comprimento de 11 bits. Esses bits são transmitidos na ordem de ID-28 (mais significativo) até ID-18 (menos significativo). Já no formato estendido, é formado por duas parcelas totalizando 29 bits (ID-28 ... ID-0).

RTR (*Remote Transmition Request*): Tanto nas mensagens padrão quanto nas estendidas, o RTR está na última posição do campo de arbitragem e seu valor deve ser dominante para mensagens de dados e recessivo para mensagens remotas.

SRR (*Substitute Remote Request*): Esse bit só está presente nas mensagens estendidas e seu valor é sempre recessivo.

IDE (*Identifier Extension Bit*): Indica qual dos formatos está sendo utilizado. No formato estendido pertence ao campo de arbitragem e é sempre recessivo. Já no formato padrão pertence ao campo de controle e é sempre dominante.

Campo de Controle: Consiste em seis bits e é formado por campos diferentes no formato padrão e no estendido. No padrão, inclui o DLC, o bit IDE e o bit reservado r1. No estendido, inclui o DLC e os dois bits reservados r1 e r0.

DLC (*Data Length Code*): Contém o número de bytes do campo de dados, sendo formado por quatro bits.

Campo de Dados: Consiste nos dados que devem ser transmitidos pela mensagem de dados. Ele pode conter entre 0 e 8 bytes, sendo o mais significativo transmitido por primeiro.

Campo CRC: Contém a seqüência CRC e o delimitador CRC. A seqüência CRC, usada para conferência da integridade da mensagem, é derivada do CRC (*Cyclic Redundancy Code*), porém, melhor adaptada para mensagens com menos de 127 bits. O delimitador CRC é constituído por um único bit recessivo.

Campo ACK (*Acknowledgement*): Possui dois bits, o trecho ACK e o delimitador ACK. Uma estação transmissora envia dois bits recessivos no campo ACK. A estação que recebe corretamente a mensagem envia a confirmação colocando um bit dominante no trecho ACK. O delimitador ACK é sempre um bit recessivo.

EOF (*End Of Frame*): Todas as mensagens, tanto mensagens de dados quanto mensagens remotas são delimitadas por uma seqüência de sete bits recessivos.

3.4.2 Mensagem Remota

É transmitida pelo barramento para requisitar a transmissão de uma mensagem de dados. A estação receptora pode a partir daí iniciar a transmissão dos respectivos dados. A mensagem remota existe tanto no formato padrão quanto no estendido e em ambos os casos é formada por seis campos de bits diferentes: SOF, campo de arbitragem, campo de controle, campo CRC, campo ACK e EOF. Ao contrário das mensagens de dados, o bit RTR é sempre recessivo e não há campo de dados.

3.4.3 Mensagem de Erro

As mensagens de erro podem ser transmitidas por qualquer estação que detecte um erro no barramento, funcionando como um alarme de falha. Consistem em dois campos diferentes, sendo o primeiro campo formado pelos sinalizadores de erro, provenientes de diferentes estações e o segundo campo formado pelo delimitador de erro.

3.4.4 Mensagem de Sobrecarga

As mensagens de sobrecarga são usadas para avisar sobre um atraso extra entre a mensagem anterior e a próxima mensagem, sejam elas mensagens de dados ou remotas. Consiste em dois campos diferentes de bits, o sinalizador de sobrecarga e o delimitador de sobrecarga. É importante principalmente em sistemas onde se deseja determinismo temporal na aplicação alvo.

4 DEVICENET

DeviceNet é uma rede industrial baseada no modelo de comunicação produtor-consumidor, suportando hierarquias de comunicação e prioridade de mensagens. Também segue as especificações do modelo OSI para redes de comunicação. Se baseia na especificação CAN, mas possui especificações adicionais para as camadas física e de enlace, além das camadas de rede e transporte. O barramento DeviceNet adota definições comuns a outros protocolos para a camada de aplicação e foi desenvolvida principalmente para conexão de dispositivos industriais simples (sensores e atuadores) com dispositivos de mais alto nível (controladores) (MACKAY et. al., 2004).

4.1 Origem

No início dos anos 90, uma empresa americana de engenharia, chamada Cincinnati Milacron, formou uma parceria com as empresas Allen-Bradley e Honeywell para o desenvolvimento de um meio de comunicação de mais alto nível baseado no protocolo CAN. Porém, o grupo se desfez e somente a empresa Allen-Bradley resolveu levar a idéia adiante de forma independente. Sua solução objetivava principalmente a comunicação entre dispositivos de campo (sensores, atuadores, etc.) e controladores industriais, passando a empregar o nome de DeviceNet (CiA, 2007a).

Em 1994 é formada a ODVA (*Open DeviceNet Vendor Association Inc.*), entidade responsável por regulamentar e regular as especificações DeviceNet, de forma a garantir compatibilidade de produtos com o padrão proposto e oferecer assistência técnica para desenvolvedores que desejem implementar soluções DeviceNet. As especificações referentes aos produtos DeviceNet são abertas, mas exige-se a realização de testes de conformidade a fim de que se obtenha a certificação de compatibilidade de qualquer produto com as normas do barramento.

4.2 Meio Físico

DeviceNet utiliza a topologia de rede em barramento, usando um cabo com malha de aterramento e quatro vias separadas em dois pares, um par para alimentação e o outro para tráfego de sinais. Há também o cabo plano, onde as vias estão dispostas em paralelo sem malha de aterramento, solução mais barata, porém menos imune à interferência eletromagnética externa. Na figura 9 pode-se observar uma representação dos cabos plano e

convencional, com suas respectivas cores para cada sinal, sendo: V+ = vermelho; V- = preto; CAN_H = branco; CAN_L = azul.

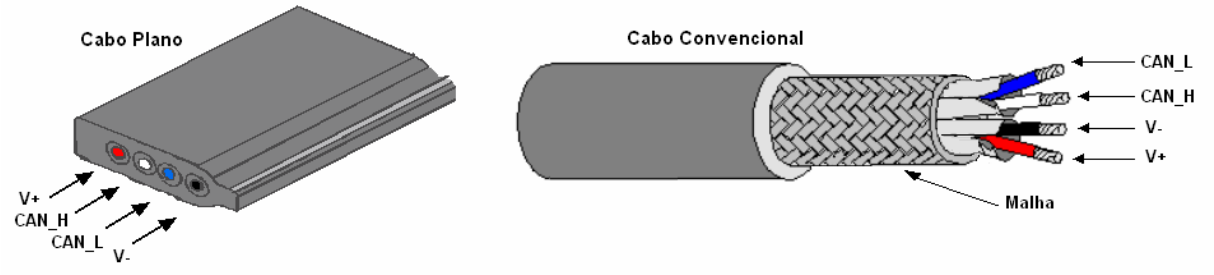


Figura 9 Cabo plano e convencional específicos para DeviceNet.

Os cabos convencionais citado podem ser de dois tipos: fino ou espesso. Nas implementações industriais, normalmente existe um cabo principal que percorre toda a planta onde se encontram os diferentes dispositivos, que é chamado de cabo tronco. Este cabo normalmente é implementado usando-se o modelo espesso. A conexão entre os dispositivos e o cabo tronco é chamada de derivação e normalmente usa-se o modelo fino. Tal disposição está representada na figura 9, onde há um exemplo de três estações ligadas ao cabo tronco através de seus cabos de derivação (ODVA, 2007c).

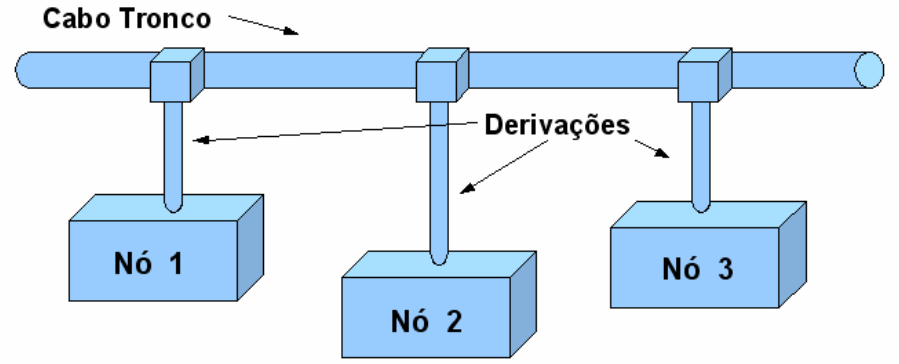


Figura 10 Implementação típica com cabo tronco e derivações.

Conforme já comentado, existem sempre limitações físicas para qualquer implementação prática de redes, que dependem de uma relação entre distância, taxa de comunicação e tipo de cabo, sendo que as taxas de transferência de dados configuráveis para uma rede DeviceNet são 125 kbit/s, 250 kbit/s ou 500 kbit/s. Na tabela 1 é apresentada uma tabela com os respectivos valores.

Tabela 1 Relação entre taxa de comunicação, tipo de cabo e comprimento.

Velocidade	Comprimento máximo do cabo tronco			Somatório de todas as derivações
	Cabo Plano	Cabo Espesso	Cabo Fino	Todos os Cabos
125 kbits/s	420 m	500 m	100 m	156 m
250 kbits/s	200 m	250 m	100 m	78 m
500 kbits/s	75 m	100 m	100 m	39 m

Redes DeviceNet possuem alimentação no próprio barramento, permitindo que dispositivos com baixo consumo de energia possam ser alimentados diretamente pela rede. Porém, a máxima corrente que pode circular pelo barramento é 8 A, fazendo com que seja necessária a divisão do barramento em duas ou mais partes com fontes de alimentação diferentes, caso o consumo seja maior que o permitido.

4.3 Enlace de Dados

Cada dispositivo é definido como um nó na rede DeviceNet, seja ele um sensor, atuador, controlador ou outro dispositivo qualquer. As mensagens usadas em uma rede DeviceNet adotam o formato padrão CAN, com 11 bits no identificador, sendo que o formato estendido (identificador de 27 bits) não é usado pela DeviceNet. A transferência de dados ocorre através do uso das mensagens de dados do CAN, sendo que as demais mensagens, como remota, de erro e sobrecarga não são usadas (ODVA, 2007b).

A DeviceNet é uma rede baseada em conexões, as quais proporcionam um caminho para troca de dados entre dispositivos. Somente quando uma conexão é estabelecida entre os dispositivos (produtor e consumidor), as transmissões associadas a ela podem ser realizadas. O identificador de conexão está inserido no campo do identificador de mensagem DeviceNet, que é a combinação do identificador MAC do dispositivo com o identificador de mensagem.

Identificador de mensagem é aquele que identifica a mensagem inserida em determinado grupo de mensagens em uma aplicação, facilitando o estabelecimento de múltiplas conexões dentro de um único grupo de mensagens.

Já os identificadores MAC são valores inteiros decimais de 0 a 63, número máximo de dispositivos suportados em uma rede. A definição do valor do identificador MAC de cada dispositivo é determinado pelo profissional responsável por implementar a rede e deve ser único para cada elemento físico presente. Dependendo do fabricante, o identificador MAC pode ser configurado por chaves seletoras existentes no corpo do dispositivo ou via softwares de configuração.

Os 11 bits do campo do identificador CAN disponíveis nas mensagens DeviceNet são subdivididos em quatro grupos de mensagens. Os grupos de mensagens foram definidos com a intenção de apresentar uma solução na qual a prioridade de acesso ao barramento não é baseada somente no identificador MAC do nó. A tabela 2 mostra as alocações DeviceNet dentro dos 11 bits do identificador CAN (ODVA, 2007d).

Tabela 2 Disposição dos campos do identificador de conexão DeviceNet.

Bits do Identificador											Campo em Hex.	Uso da Identificação
10	9	8	7	6	5	4	3	2	1	0		
0	Grupo 1				MAC ID						000 – 3ff	Grupo de Mensagens 1
1	0	MAC ID						Grupo 2			400 – 5ff	Grupo de Mensagens 2
1	1	Grupo 3			MAC ID						600 – 7df	Grupo de Mensagens 3
1	1	1	1	1	Grupo 4						7e0 – 7ef	Grupo de Mensagens 4
1	1	1	1	1	1	1	X	X	X	X	7f0 – 7ff	Identificador CAN inválido

Tanto mensagens explícitas de conexão quanto mensagens de I/O podem ser estabelecidas pelos grupos de mensagem 1, 2 e 3 enquanto que mensagens de administração do sistema utilizam-se do grupo 4. Conforme o esquema de controle de acesso ao meio definido pelo CAN, mensagens do grupo 1 possuem maior prioridade que mensagens do grupo 2, que por sua vez possuem maior prioridade que mensagens do grupo 3. A estratégia de utilização de cada grupo é a seguinte:

Grupo 1 - Nas transmissões de mensagens do grupo 1, a prioridade de acesso ao barramento é uniformemente distribuída entre os dispositivos da rede. Quando duas ou mais mensagens do grupo 1 competem pelo acesso ao barramento CAN, a mensagem com valor numérico mais baixo no identificador de mensagem vence a comparação e ganha o acesso. Caso duas mensagens possuam o mesmo identificador de mensagem, o dispositivo com menor valor do identificador MAC terá acesso ao meio.

Grupo 2 - DeviceNet pré-define um conjunto de conexões para facilitar a comunicação existente em uma aplicação mestre/escravo. O identificador de mensagem 7 é reservado para uso na detecção de nós que possuam identificadores MAC idênticos. No grupo 2, o identificador MAC pode tanto ser do nó de origem quanto do nó de destino da mensagem. Quando se estabelece uma conexão através do grupo 2, a aplicação determina onde o identificador MAC de origem ou de destino deve ser especificado. Quando duas ou mais transmissões do grupo 2 competem pelo acesso ao barramento, a mensagem que possuir o valor inferior no identificador MAC irá ganhar o acesso ao meio. Caso o identificador MAC seja idêntico para duas ou mais mensagens, aquela que possuir o identificador de mensagens com menor valor irá ganhar o acesso ao meio.

Grupo 3 - As mensagens do grupo 3 possuem alguns valores de identificadores de mensagens reservados. O identificador de mensagem 5 é usado para o envio de respostas associadas com uma pergunta de mensagem explícita, assim como indicações de estado e retirada de dispositivo. O identificador de mensagem 6 é usado para o envio de requisição de mensagem explícita desconectada. O identificador de mensagem 7 é inválido e não é usado. Mensagens que estabelecem dinamicamente conexões de mensagem explícitas são transmitidas dentro do grupo 3 e reservam o valor 6 (perguntas) ou 5 (respostas) como identificador de mensagem do grupo 3. Essas mensagens são referenciadas como mensagens explícitas desconectadas. Mensagens explícitas desconectadas são processadas pelo gerenciador de mensagens desconectadas (UCMM).

Grupo 4 - Os identificadores que não estão reservados devem ser usados para propósitos de administração do sistema. Suas mensagens disponibilizam uma ferramenta para recuperação de nós que tenham ficado *offline* por terem o mesmo endereço de outro nó na rede.

Uma rede DeviceNet normalmente possui um dispositivo definido como mestre (geralmente um CLP) e demais dispositivos definidos como escravos, sendo que estes podem ser produtores, consumidores ou os dois. O mestre é o dispositivo que recolhe e distribui dados de I/O para o controle do processo. Escravos são os dispositivos dos quais o mestre recolhe os dados de I/O e para os quais o mestre distribui os dados de I/O. Neste conceito, um dispositivo produtor, normalmente observa requisições ou o tempo de amostragem e produz valores de variáveis no barramento. Já um consumidor, apenas aguarda mensagens ou faz requisições e consome os valores publicados pelo produtor.

As implementações DeviceNet aceitam variações desse modelo, adotando três tipos de mensagens que são configuráveis via software. Entretanto, nem todos os equipamentos aceitam as três formas de troca de dados, deixando disponível ao usuário apenas uma ou duas delas. Os tipos de mensagens que aparecem disponíveis nos softwares de configuração são as de pesquisa, as cíclicas e as de mudança de estado, cujas descrições são as seguintes:

Pesquisa (*Poll*) – Uma estação configurada para operar com mensagens de pesquisa receberá dados de forma seqüencial conforme a lista de varredura do dispositivo mestre. A freqüência com que o mestre envia mensagens de pesquisa é determinada pelo número de nós na lista de varredura, pela taxa de comunicação configurada para a rede, pelo tamanho das mensagens produzidas e pelo tempo de processamento do dispositivo mestre. Esse método de envio de mensagens pode proporcionar um comportamento determinístico para o sistema.

Cíclica (*Cyclic*) – Uma estação configurada para operar com mensagens cíclicas produz seus dados em um intervalo de tempo definido previamente. As mensagens cíclicas permitem ao usuário configurar o sistema para produzir dados com uma freqüência apropriada para determinada aplicação, que dependendo do caso pode reduzir o tráfego no barramento e usar de forma mais eficiente a largura de banda disponível. Também permite, quando configurado adequadamente, um comportamento determinístico para o sistema.

Mudança de estado (*Change of State*) – Uma estação configurada para operar com mensagens de mudança de estado produz as mensagens sempre que os valores referentes ao conteúdo das mesmas forem alterados. Porém, para garantir que o dispositivo continua operando de maneira correta, ele pode ser configurado para enviar essas mensagens com uma taxa de envio ajustável, de forma a permitir a sinalização de que continua ativo, mesmo sem mudanças nos valores medidos. Da mesma forma, as mensagens de mudança de estado também podem ser configuradas para uma taxa de envio limite, evitando que utilizem o barramento de forma desnecessária.

Cada uma destas formas de trocas de dados citadas acima possui suas próprias características, mas na prática as mensagens de pesquisa são as mais utilizadas, por apresentarem maior facilidade no momento da implementação. Porém, com a utilização de um grande número de dispositivos na mesma rede, esse tipo de mensagem pode prejudicar o tráfego de informação, pois ocupa a banda disponível com informações muitas vezes repetidas. Uma solução para deste problema é o uso de mensagens de mudança de estado, que só enviariam os dados quando estes fossem alterados, mas exige maior tempo de configuração devido a sua complexidade de uso.

4.4 Conexões Pré-definidas

Para padronizar o estabelecimento de uma conexão para transferência de dados entre dispositivos, definiu-se um conjunto de conexões, o qual facilita conexões tipicamente vistas em um relacionamento mestre/escravo. Essas conexões são referidas coletivamente como “conexões pré-definidas entre mestre e escravo” (ODVA, 2007d).

O mestre “possui” os dispositivos cujos identificadores MAC aparecem na sua lista de *scan*, a qual é examinada para determinar com qual escravo o mestre irá se comunicar. Exceto pela checagem de identificador MAC duplicado, um escravo não pode iniciar qualquer comunicação antes do mestre lhe dizer para fazer isso.

Muitos dos passos envolvidos na criação e configuração de uma conexão são trocados dentro das conexões pré-definidas entre mestre e escravo. Com isso, os meios pelos quais um ambiente de comunicação pode ser estabelecido, utilizam menos recursos da rede e dos dispositivos. Na tabela 3 constam os diferentes tipos de mensagens pré-definidas nas especificações de acordo com o campo dos identificadores CAN. Além disso, a tabela contém também os identificadores reservados para outras aplicações.

Tabela 3 Conexões Pré-definidas entre Mestre e Escravo.

Bits do Identificador											Valor Hex	Descrição
10	9	8	7	6	5	4	3	2	1	0		
											000-3FF	Mensagens do Grupo 1
0	Identificador 1 - Grupo 1											Slave's I/O Multicast Poll Response
0	Identificador 2 - Grupo 1											Slave's I/O Change of State or Cyclic Message
0	Identificador 3 - Grupo 1											Slave's I/O Bit-Strobe Response Message
0	Identificador 4 - Grupo 1											Slave's I/O Poll Response or Cyclic/COS Ack. Message
											400-5FF	Mensagens do Grupo 2
1	0	Identificador 1 - Grupo 2										Master's I/O Bit-Strobe Command Message
1	0	Identificador 2 - Grupo 2										Master's I/O Multicast Poll Group ID
1	0	Identificador 3 - Grupo 2										Master's Change of State/Cyclic Acknowledge Message
1	0	Identificador 4 - Grupo 2										Slave's Explicit/Unconnected Explicit Response Message
1	0	Identificador 5 - Grupo 2										Master's Connected Explicit Request Message
1	0	Identificador 6 - Grupo 2										Master's I/O Poll Command/Change of State/Cyclic Message
1	0	Identificador 7 - Grupo 2										Group 2 Only Unconnected Explicit Request Message
1	0	Identificador 8 - Grupo 2										Duplicate MAC ID Check Message
											600-7BF	Mensagens do Grupo 3
1	1	Identificador 1 - Grupo 3										Explicit Messaging Connection
1	1	Identificador 2 - Grupo 3										Unconnected Explicit Response (UCMM) Message
1	1	Identificador 3 - Grupo 3										Unconnected Explicit Request (UCMM) Message
											7C0-7EF	Mensagens do Grupo 4
1	1	1	1	1	Identificador 1 - Grupo 4							Reserved Group 4 Identifiers
1	1	1	1	1	Identificador 2 - Grupo 4							Communication Fault Response Message
1	1	1	1	1	Identificador 3 - Grupo 4							Communication Fault Request Message
1	1	1	1	1	Identificador 4 - Grupo 4							Offline ownership Response Message
1	1	1	1	1	Identificador 5 - Grupo 4							Offline ownership Request Message
1	1	1	1	1	Identificador 6 - Grupo 4							Invalid CAN Identifiers

Os seguintes tipos de mensagens são mencionados na tabela:

I/O Bit-Strobe Command/Response Messages - Um comando *bit-strobe* é uma mensagem de I/O transmitida por um mestre, podendo ser *multicast* (multiponto). Múltiplos escravos podem receber e reagir ao mesmo comando *bit-strobe*. A resposta *bit-strobe* é uma mensagem de I/O que um escravo transmite de volta ao mestre quando um comando *bit-strobe* é recebido. As mensagens *bit-strobe* são usadas para trocas de pequenas quantidades de dados de I/O entre um mestre e seus escravos. Um comando *bit-strobe* envia um bit de dados de saída para cada escravo cujo identificador MAC está na sua lista de *scan*. Essas mensagens contém um pacote de 64 bits (8 bytes) de dados de saída, sendo uma saída por identificador MAC na rede. Um bit é endereçado para cada identificador MAC suportado na rede (0...63).

Uma resposta *bit-strobe* retorna até 8 bytes de dados de entrada ou informações de estado de cada escravo para o mestre. Pelo fato da conexão *bit-strobe* ter sido criada para uma rápida e eficiente troca de pequenos pacotes de dados entre o mestre e seus escravos, o campo de dados da mensagem não ultrapassa os 8 bytes.

I/O Poll Command/Response Messages - Um comando *poll* é uma mensagem de I/O que é transmitida pelo mestre, direcionada para um único escravo específico (ponto-a-ponto). Um mestre deve transmitir uma mensagem de comando *poll* separada para cada um dos escravos que deve consultado. A resposta *poll* é uma mensagem de I/O que um escravo transmite de volta ao mestre após o comando *poll* ter sido recebido. Enquanto as mensagens *bit-strobe* movem pequenas quantidades de dados de I/O, as mensagens *poll* movem qualquer quantidade de dados de I/O entre mestre e seus escravos. Um comando *poll* envia qualquer quantidade de dados (fragmentado ou não) para o escravo de destino e consequentemente recebe qualquer quantidade de dados como resposta.

I/O Change of State/Cyclic Messages - As mensagens de mudança de estado/cíclicas são transmitidas pelo mestre ou pelo escravo e são direcionadas para um único nó (ponto-a-ponto). Uma mensagem de reconhecimento deve ser retornada em resposta a essa mensagem. As mensagens de mudança de estado/cíclicas movem qualquer quantidade de dados de I/O entre mestre e escravo.

I/O Multicast Poll Messages - Um comando *poll* multiponto é uma mensagem de I/O transmitida pelo mestre diretamente para um ou mais escravos. A resposta *poll* multiponto é uma mensagem de I/O que um escravo transmite de volta para o mestre quando um comando *poll* multiponto é recebido. A conexão *poll* multiponto se diferencia da conexão *poll* na sua habilidade de poder ser multiponto ou ponto-a-ponto. Qualquer número de escravos pode pertencer a um grupo multiponto do mestre e vários grupos multiponto podem existir para cada mestre. Essa forma de conexão permite um compartilhamento simultâneo de dados entre um mestre e seu grupo de escravos. Também envia qualquer quantidade de dados (fragmentados ou não) para um escravo de destino.

Explicit Response/Request Messages - Mensagens de requisição explícitas são usadas para realizar operações como ler e escrever atributos. Mensagens de resposta explícita indicam o resultado do atendimento de serviço da mensagem de requisição explícita.

Group 2 Only Unconnected Explicit Request/Response Messages - É a conexão usada para alocar/ceder as mensagens pré-definidas. Essa porta é reservada e não deve ser usada para nenhum outro propósito além de enviar mensagens de estado e desligamento de

dispositivos. Essas mensagens são transmitidas usando o mesmo identificador CAN que as mensagens de resposta explícitas.

Duplicate MAC ID Check Message - Cada dispositivo físico na DeviceNet deve ser implementado com um único identificador MAC, cuja configuração envolve a intervenção do usuário. Pelo fato do identificador MAC estar envolvido na definição do significado da transmissão DeviceNet, todos os módulos DeviceNet são requeridos a participar da detecção de identificador MAC duplicado. A mensagem de checagem deve ser transmitida duas vezes consecutivas e depois da transmissão um módulo espera no mínimo 1 segundo pela resposta antes de tomar as ações apropriadas definidas para ele.

UCMM - O gerenciador de mensagens desconectadas (UCMM) proporciona o estabelecimento de conexões de mensagens explícitas, processando dois tipos de serviços. Um serviço é a abertura de conexão de mensagem explícita (serviço 4Bh), usado para estabelecer uma conexão. O outro serviço é o fechamento de conexão (serviço 4Ch), usado para excluir um objeto de conexão e todos os recursos associados. Esses serviços são acessados usando as perguntas e respostas explícitas desconectadas.

Explicit Messaging Connection - São mensagens transmitidas através de uma conexão de mensagem explícita. Após a abertura de uma conexão de mensagens explícitas, os dispositivos envolvidos podem realizar a troca de dados, que são parâmetros de serviços específicos para um tipo particular de aplicação.

Reserved Group 4 Identifiers – Formam um grupo de identificadores reservados para futuras aplicações.

Communication Fault Response/Request - O identificador de mensagem 2Ch é usado por nós com falha na comunicação quando produzem mensagens de resposta de falha de comunicação. O identificador de mensagem 2Dh é usado por ferramentas para produzir mensagens de pergunta de falha de comunicação.

Offline ownership Response/Request – O identificador de mensagem 2Eh é usado por ferramentas para produzir mensagens de resposta de propriedade *offline*. O identificador de mensagem 2Fh é usado em mensagens de configuração de conexão *offline*.

Invalid CAN Identifiers – São identificadores CAN inválidos e não são utilizados.

4.5 Camadas Superiores

A DeviceNet usa o CIP (*Commom Industrial Protocol*), como especificação de funcionamento das camadas de mais alto nível, que é o mesmo usado por outras redes industriais, como ControlNet e EtherNet/IP. O CIP é um protocolo orientado a objetos, onde cada objeto possui atributos (dados) e serviços (comandos), os quais apresentam determinado comportamento (reação a eventos). A especificação CIP define dois tipos diferentes de objetos: os de comunicação e os de aplicação. Para um determinado tipo de dispositivo, já é previamente definido um conjunto mínimo de objetos comuns, implementados por todos os desenvolvedores daquele mesmo tipo de equipamento. Com isso, é garantida uma interoperabilidade entre dispositivos independente do fabricante, facilitando a implementação de um sistema por parte do usuário (ODVA, 2007b).

Outra vantagem do uso do protocolo CIP para os níveis mais altos, é que o programador da aplicação não precisa saber qual rede detém o dispositivo (DeviceNet, ControlNet ou EtherNet/IP), pois seus objetos de aplicação comum possuem o mesmo significado, independente do meio.

O conjunto mínimo de objetos comuns para determinado tipo de equipamento, citado anteriormente, está presente no perfil dos dispositivos, o qual inclui opções de configuração e formato de dados. Assim sendo, todos os dispositivos que seguem um perfil padrão terão os mesmos formatos nos dados de entrada e saída, responderão aos mesmos comandos e apresentarão as mesmas opções de configuração e comportamento que outros dispositivos que seguem o mesmo perfil. Entretanto, as especificações DeviceNet também permitem que desenvolvedores adicionem suas próprias características ao perfil mínimo requerido, desde que essas características adicionais sejam implementadas em concordância com as normas.

Para configuração da rede o barramento DeviceNet utiliza arquivos EDS (*Electronic Data Sheet*), que acompanham cada produto DeviceNet. O arquivo EDS é um simples arquivo de texto, mas contém todas as informações referentes ao produto, como tipo de dispositivo, parâmetros de configuração, fabricante, versão do produto e demais características. Estes arquivos são usados por ferramentas de configuração de rede para identificar e configurar automaticamente produtos quando adicionados ao projeto.

5 TECNOLOGIAS DISPONÍVEIS

Para o desenvolvimento do presente trabalho, se mostra indispensável a realização de um levantamento das soluções disponíveis para monitoração e análise de barramento industriais. Para tanto, são apresentados trabalhos acadêmicos envolvendo, mesmo que de forma indireta, o assunto, assim como soluções comerciais atualmente disponíveis no mercado.

5.1 Trabalhos Acadêmicos

Ao longo dos últimos anos, devido ao significativo aumento no uso de redes de comunicação em ambientes industriais, alguns trabalhos acadêmicos foram divulgados envolvendo o assunto. Não foram localizados trabalhos tratando especificamente do monitoramento de redes DeviceNet, mas sim da análise de outros barramentos similares, deixando evidente a importância de ferramentas deste tipo.

Como exemplo, pode-se citar o trabalho de desenvolvimento da ferramenta BR-Tool (*Bus Real-Time Validation and Monitoring Tool*), para validação temporal de barramentos industriais. Tomando por base a norma IEC61158, que especifica oito diferentes redes de comunicação para aplicação em ambientes industriais, Husemannn (2003) desenvolve uma ferramenta capaz de validar temporalmente aplicações que usem esta mesma especificação. No trabalho apresenta-se a utilização da ferramenta sobre dois barramentos industriais: Profibus e Foundation Fieldbus, além de um terceiro, o CanOpen, que apesar de não estar na norma IEC61158 também é largamente aplicado na indústria. O barramento CanOpen parte do mesmo princípio do DeviceNet, por tomar como base o protocolo CAN.

O BR-Tool é apresentado como um conjunto de módulos com diferentes funções. A partir da captura de eventos no barramento, ocorre a marcação temporal, seguida pelo registro e filtragem da informação. Em seguida, é feita a validação temporal, que visa identificar se as informações coletadas no barramento atendem aos requisitos temporais previamente definidos pelo usuário, e por fim ocorre a exibição dos dados ao usuário. Essas tarefas são realizadas com a utilização de uma placa também desenvolvida por Husemann (2007), implementada com lógica programável (FPGA) e um microcontrolador de 32 bits, respectivamente para interface entre o barramento e um computador.

No final do trabalho são apresentados estudos de caso, onde a ferramenta foi utilizada em diferentes aplicações industriais reais, constatando casos onde implementações inadequadas das redes podem causar sérios problemas, como o envio de dados fora do

período de tempo esperado e conseqüente funcionamento inadequado do sistema (HUSEMANN, 2007).

Outro trabalho de significativa relevância trata da comparação dos barramentos DeviceNet, ControlNet e EtherNet/IP sob vários aspectos. O objetivo dos autores era analisar como cada rede de comunicação poderia ser utilizada em uma aplicação com sensores, atuadores e controladores. Para cada protocolo foram estudados parâmetros durante diversas situações de controle, como utilização de banda disponível e características de atraso de tempo. De posse destes dados, são apresentadas as vantagens e desvantagens de cada um dos tipos de rede analisados (LIAN, 2001).

Como vantagens da rede DeviceNet, são citados justamente aspectos já mencionados no capítulo 3, como o fato do CAN poder ser um protocolo determinístico, as mensagens possuírem um campo específico para determinação de prioridade e as mensagens com maior prioridade sempre ganharem a disputa pelo acesso ao barramento sem causar destruição do pacote de mensagem. Também são levantados alguns pontos que os autores consideram desvantagens, como a máxima taxa de transmissão de 500 kbit/s.

Lian (2001) realizou alguns estudos de caso simulando o comportamento de diferentes configurações de rede utilizando o software MatLab, mas sempre usando mensagens para envio de oito bytes de dados. Como resultado, observou-se que para aplicações onde exigem-se determinismo temporal e mensagens com maior prioridade sobre as demais, o protocolo DeviceNet é o mais indicado.

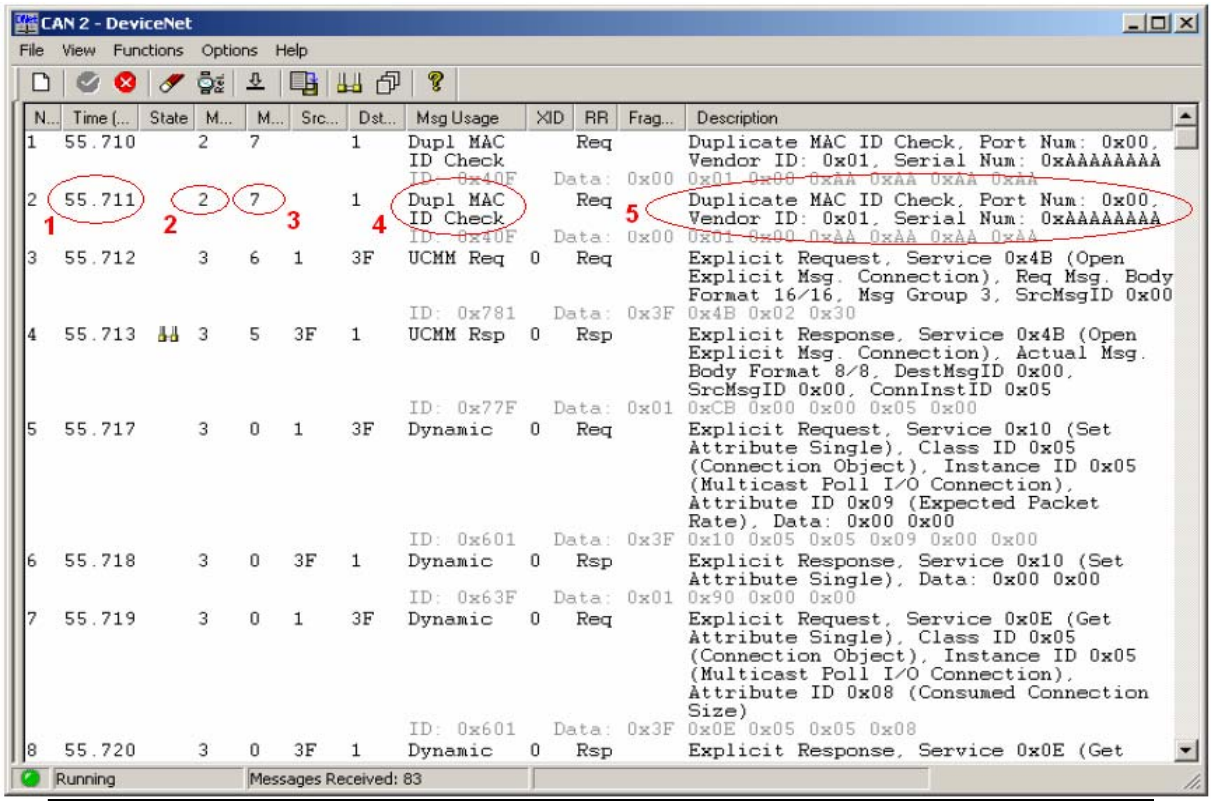
5.2 Soluções Comerciais

A ODVA, por ser responsável pela normatização de tecnologias de redes baseadas no CIP, apresenta em seu site uma lista de fabricantes de ferramentas de configuração e diagnóstico de redes DeviceNet. Tal lista foi utilizada no presente trabalho como guia para localizar os fabricantes que disponibilizam ferramentas semelhantes ao sistema proposto. Dentre as empresas citadas estão a IXXAT Automation GmbH e a Woodhead Industries, mas há outros desenvolvedores que também disponibilizam soluções em monitoramento de redes DeviceNet, como a Vector Informatik. Com base nos dados disponíveis, faz-se aqui uma pequena apresentação das características dos produtos de cada um, que serviu de base para especificação da proposta (ODVA, 2007a).

5.2.1 IXXAT

A IXXAT Automation GmbH desenvolve placas para interface de computadores com redes CAN, assim como softwares para monitoramento e análise do tráfego de dados. Há várias opções de formas de aquisição dos dados, dentre as quais podem-se citar placas PCI e PCIe, conversores USB e Bluetooth e cartões com interface PCMCIA. O software oferecido pela empresa para tratamento das mensagens CAN se chama canAnalyser, mas para que as mensagens sejam interpretadas dentro das especificações DeviceNet, é necessária a aquisição de um software adicional, o DeviceNet Module. A empresa oferece uma versão demonstrativa do canAnalyser, mas não do seu módulo DeviceNet. Sendo assim os dados aqui apresentados foram baseados nos manuais da ferramenta conforme disponibilizados no site da empresa (IXXAT, 2007).

Segundo o fabricante, o módulo DeviceNet interpreta as mensagens de acordo com a sua especificação, exibindo parâmetros discriminadamente, como grupo de mensagem, identificador MAC e identificador de mensagem. As informações recebidas são divididas em mensagem desconectada, mensagem explícita, mensagem de entrada/saída, mensagem de checagem de MAC, mensagem de taxa de transferência, entre outras. Pode-se ainda habilitar a filtragem de dados, para exibir e também armazenar somente o que for selecionado pelo operador, que pode ser através do identificador MAC ou do tipo da mensagem. Os dados armazenados podem ser exportados para arquivos de texto, facilitando a utilização por outros softwares. Abaixo, na figura 11, está uma tela do software de análise de redes DeviceNet citado, onde aparecem itens como: momento da aquisição, grupo, identificador, MAC, utilização da mensagem e uma descrição correspondente.



1 – Momento da aquisição
2 – Grupo da mensagem
3 – Identificador da mensagem
4 – Utilização da mensagem
5 – Descrição da mensagem

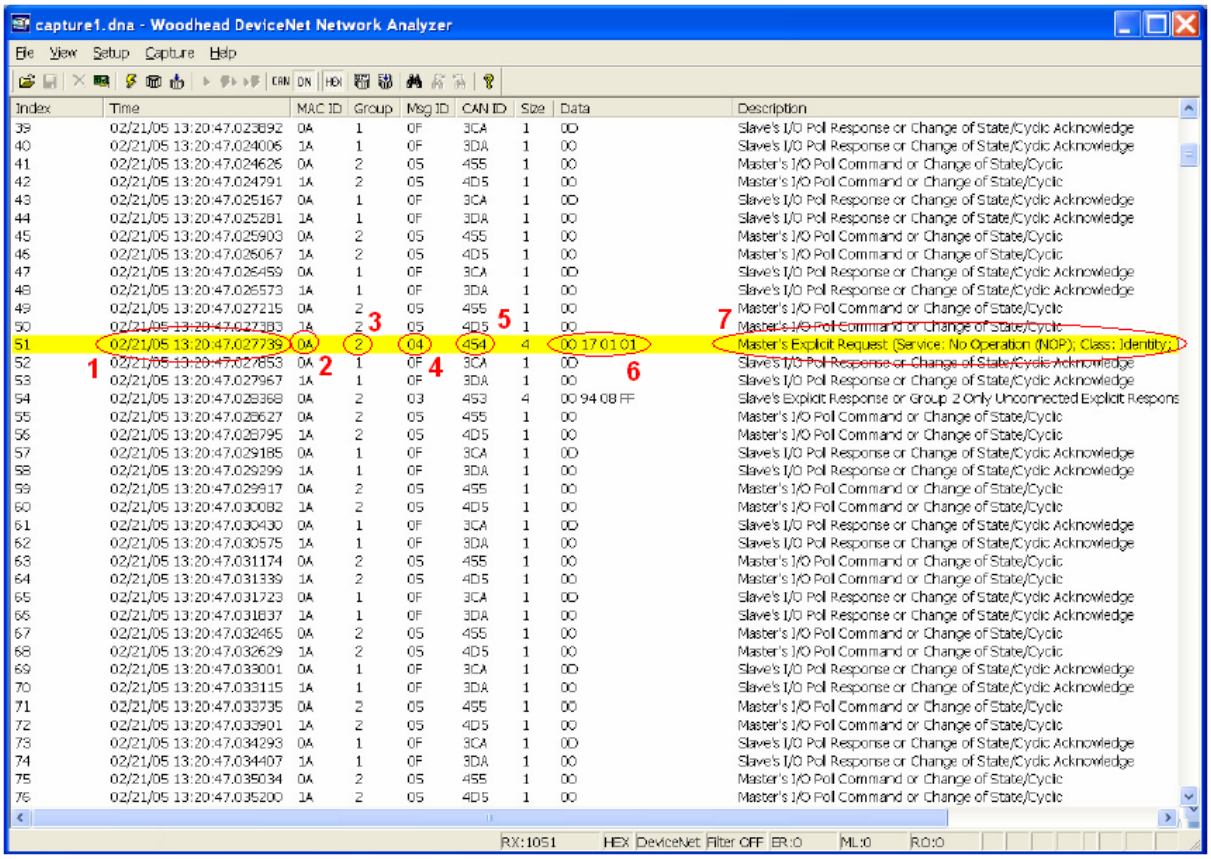
Figura 11 Módulo DeviceNet da IXXAT.

5.2.2 Woodhead

A Woodhead Industries é outra empresa que desenvolve placas para interface de computadores com redes DeviceNet. As placas são disponibilizadas para interfaces ISA, PCI e PC/104 e também cartões PCMCIA. Para análise de dados, oferece um software chamado DeviceNet Network Analyzer. Tal software não possui versão de demonstração, impossibilitando uma análise mais detalhada do produto (WOODHEAD, 2007).

Como características do software pode-se citar a captura das mensagens, exibição e filtragem. A ferramenta pode selecionar, por exemplo, o tempo em que ocorreu a aquisição, o identificador MAC, grupo da mensagem e identificador de mensagem, assim como os valores do campo de dados. Oferece a opção de captura de até 100 milhões de mensagens em um período máximo de 46 dias, diagnosticando eventuais problemas sem interferir no barramento. A ferramenta permite definir exatamente o ponto de início da captura, baseado em mensagens específicas, assim como o número de mensagens que deve adquirir. Na figura

12 está presente uma tela do DeviceNet Network Analyzer, onde pode-se observar a exibição de dados como momento da aquisição, identificador MAC, grupo da mensagem, identificador da mensagem, identificador CAN, valor dos dados e uma descrição da mensagem.



1 – Momento da aquisição
2 – Identificador MAC
3 – Grupo da mensagem
4 – Identificador da mensagem
5 – Identificador CAN
6 – Valor do campo de dados
7 – Descrição da mensagem

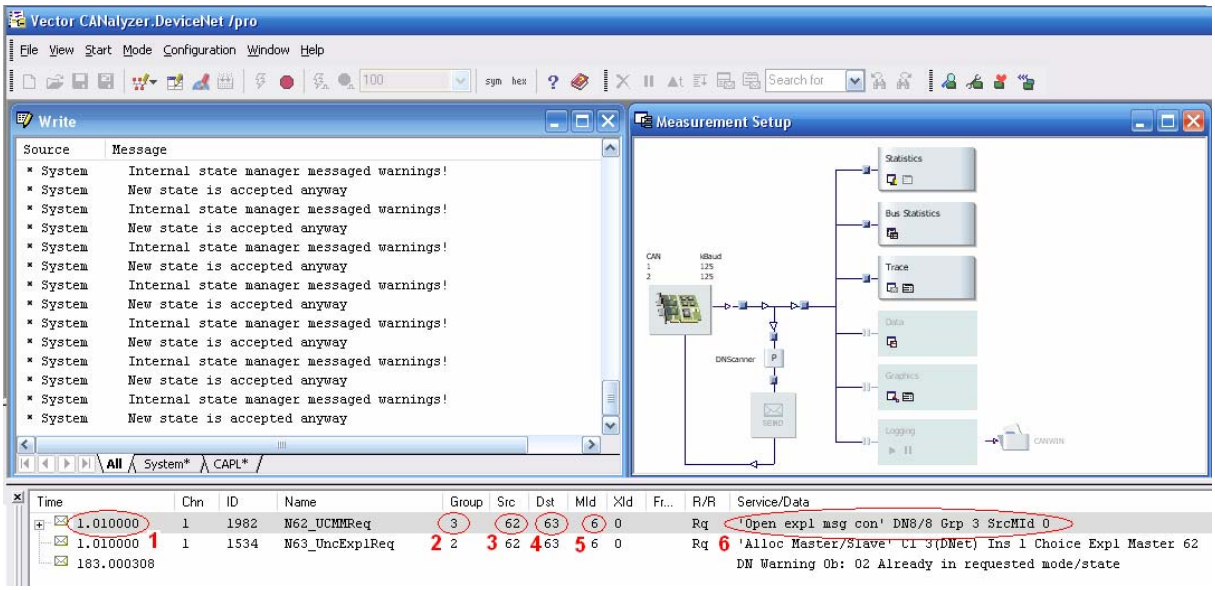
Figura 12 DeviceNet Network Analyzer da Woodhead.

5.2.3 Vector

A Vector Informatik também oferece soluções para o monitoramento de tráfego em redes DeviceNet. Dentre as opções de interface com o computador, oferece placas PCI, cartões PCMCIA e conversores USB. O software base para aquisição de mensagens CAN é o CANalyzer, mas para a interpretação de mensagens DeviceNet é preciso um pacote adicional, chamado CANalyzer.DeviceNet. Tanto o CANalyzer como o seu módulo adicional CANalyzer.DeviceNet possuem versões demonstrativas disponíveis para download, porém,

por se tratar de uma simulação, os dados gerados pelo analisador podem não fazer sentido, conforme informado pelo próprio fabricante (VECTOR, 2007).

Tal software apresenta gráficos com o fluxo de dados de todos os dispositivos presentes na rede. Monitora a taxa de transferência de dados e lê os atributos cíclicos das estações. Pode capturar mensagens geradas isoladamente e identificar seu conteúdo. Na figura 13 está presente uma imagem da tela de operação onde podem ser vistos campos como identificador, grupo, origem da mensagem, destino da mensagem e descrição da mensagem.



1 – Momento da aquisição	**4** – Destino da mensagem
2 – Grupo da mensagem	**5** – Identificador da mensagem
3 – Origem da mensagem	**6** – Descrição da mensagem

Figura 13 CANalyzer.DeviceNet da Vector.

5.2.4 Comparação entre fabricantes

A partir dos dados disponibilizados pelos fabricantes das ferramentas acima citadas, e até mesmo das versões demonstrativas disponíveis, pode-se montar um quadro comparativo, onde constam as principais características de cada produto (Tabela 4).

De forma geral, percebe-se que os três produtos possuem praticamente as mesmas características. O que difere um do outro, além é obvio da interface de operação, é a opção de exibição de gráficos, recurso este somente disponibilizado pelo software da Vector. Um dos fabricantes, a Woodhead, não apresenta a indicação de mensagens fragmentadas, como os outros fazem. Demais características, como exibição do identificador de mensagem,

identificador MAC, grupo de mensagem, valor do campo de dados, filtros e armazenamento em arquivos de texto, são comuns a todos.

Tabela 4 Comparação entre fabricantes.

	Fabricante		
Funções	Ixxat	Woodhead	Vector
Identificador da mensagem	Sim	Sim	Sim
Identificador MAC	Sim	Sim	Sim
Grupo da mensagem	Sim	Sim	Sim
Origem	Sim	Sim	Sim
Destino	Sim	Sim	Sim
Valor dos dados	Sim	Sim	Sim
Descrição da mensagem	Sim	Sim	Sim
Fragmentação	Sim	Não	Sim
Gráficos	Não	Não	Sim
Filtros	Sim	Sim	Sim
Salva em arquivo texto	Sim	Sim	Sim
Demo disponível	Não	Não	Sim

6 SISTEMA DE MONITORAÇÃO E IDENTIFICAÇÃO

Com base no que foi apresentado até aqui, pode-se perceber a atual importância das redes de comunicação industriais no campo da automação industrial e também a forma como se destaca entre elas o protocolo CAN e mais especificamente o padrão de rede DeviceNet.

O barramento DeviceNet, ao contrário de outros barramentos industriais, tem sua performance bastante dependente da configuração escolhida pelo profissional integrador, que precisa adequar as demandas do projeto e prioridade de mensagens utilizadas. A observação do funcionamento da rede em condições reais de operação é uma das formas de se validar o funcionamento da aplicação desejada. Esta forma de validação é evidenciada claramente pela disponibilidade de diferentes ferramentas no mercado desenvolvidas especificamente para este fim.

Neste cenário, optou-se pelo desenvolvimento de uma ferramenta de monitoração e identificação de dados para redes industriais padrão DeviceNet. O sistema consiste de uma placa para interface entre o barramento CAN, permitindo captura de mensagens da rede em tempo de execução, e um programa no computador que exibe as mensagens adquiridas. Na figura 14 está uma visão geral do sistema, que consiste basicamente em conectar a placa de interface diretamente à rede DeviceNet e conectar o computador à placa através da porta USB.

A placa de aquisição captura os dados de barramento de forma transparente aos dispositivos existentes na rede, armazenando em um *buffer* as mensagens DeviceNet capturadas. Essas mensagens são enviadas ao computador pela porta USB e tratadas pelo softwares de interface com o usuário, que depois de identificar e separar cada campo de informações da mensagem, exibe os seguintes dados na tela: tempo, identificador, grupo, MAC ID, tamanho, dados, significado.

Sua importância se destaca tanto na área acadêmica quanto profissional. Ou seja, a existência desse tipo de ferramenta é importante tanto para alunos, que estejam estudando ou montando experiências com produtos DeviceNet, quanto para profissionais de automação que queiram comprovar o funcionamento de sua aplicação, bem como identificar algum tipo de problema no barramento.

Como demonstrado na seção 5, existem diversas soluções comerciais de hardware e ferramentas de apoio, para auxiliar desenvolvedores de aplicações DeviceNet. Entretanto, são soluções fechadas. Também serviu de motivação o fato de não se encontrar versão compatível com língua portuguesa, assim como qualquer ferramenta desenvolvida no Brasil.

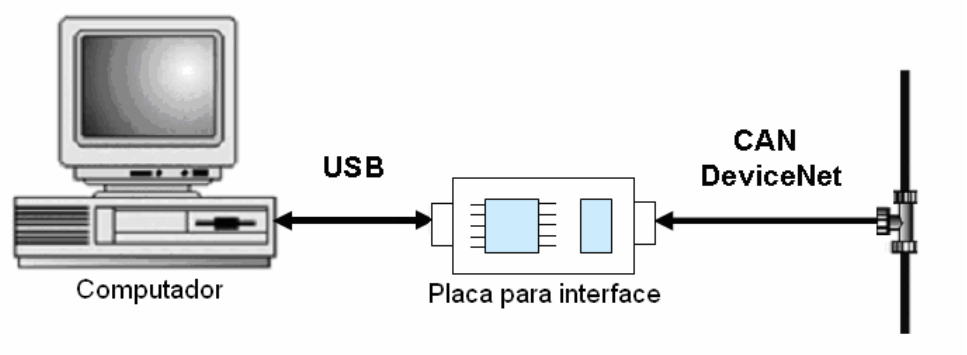


Figura 14 Visão geral do sistema proposto.

A conexão do sistema e a captura das mensagens no barramento ocorrem de forma transparente aos dispositivos existentes na rede. É justamente pelo fato de ser uma ferramenta de monitoração do tráfego de dados que deve-se manter a rede implementada com sua configuração original de funcionamento. Se o sistema proposto fosse adicionado ao barramento como mais um dispositivo ativo, alteraria a configuração existente, o que não é desejado.

6.1 Hardware

Para o desenvolvimento proposto procurou-se alguma solução comercial disponível na forma de kit de desenvolvimento ou plataforma de avaliação que pudesse ser usada. Selecionou-se para tal um kit de desenvolvimento produzido pela empresa Infineon Technologies. Optou-se por esta abordagem, a fim de simplificar o desenvolvimento do projeto.

O kit selecionado (XC164CM U CAN Start Kit) possui um microcontrolador embarcado que inclui dois canais de comunicação CAN, além de 10 pinos de entrada e saída para uso geral e um conector USB para comunicação com o computador. Observa-se que o "XC164CM U CAN Start Kit" é relativamente pequeno (aproximadamente 76 x 34 x 13 mm - comprimento x largura x altura) e com alimentação e programação através da própria porta USB. Na figura 15 apresenta-se uma imagem do kit escolhido.

Figura 15 U CAN Start Kit da Infineon.

O microcontrolador utilizado no kit é o SAK-XC164CM-8F40F da empresa Infineon Technologies, que possui unidade central de processamento de 16 bits e freqüência de até 40MHz, apesar de que no kit, o cristal oscilador utilizado ser de 6MHz. Possui 64 Kbytes de memória de código (*flash*) e 2 Kbytes de memória de dados (RAM).

Os dois canais de comunicação CAN disponíveis no kit servem para fins de simulação de um barramento. Enquanto um canal é usado para gerar mensagens, o outro canal é usado para fazer a captura das mesmas. Inicialmente, utilizou-se a opção de simulação durante os testes de funcionamento do kit. Entretanto uma rede DeviceNet real foi utilizada ao longo do desenvolvimento do trabalho, pois algumas questões práticas de implementação passam despercebidas no caso de uma simulação.

Como o sistema deve ficar conectado ao barramento de forma a não interferir no seu funcionamento, simplesmente se faz uma conexão às duas vias do canal de comunicação CAN, CAN_H e CAN_L, para observar o que está trafegando.

O kit de desenvolvimento é programado com o uso de um compilador que acompanha o mesmo, cuja comunicação é feita pela porta USB. O gerenciamento do fluxo de dados entre o kit e o computador é feito por um *driver* instalado juntamente com o compilador. Tal *driver*, denominado *Device Access Server* (DAS), simula uma porta serial no gerenciador de dispositivos do computador, permitindo assim a comunicação do kit pela USB como se fosse por uma porta serial EIA 232, rodando de forma transparente ao usuário.

É através do DAS que é possível enviar o programa compilado ao microcontrolador, assim como iniciar e parar a execução do programa. O DAS também permite fazer uma depuração dos dados no compilador e observar o andamento do software rodando no kit passo-a-passo.

Como o sistema operacional Windows não permite que dois aplicativos acessem a mesma porta de comunicação ao mesmo tempo, não é possível acessar os dados que são enviados pela porta USB no programa desenvolvido enquanto o DAS está rodando. Sendo assim, foi necessário adquirir os dados através de outra porta USB. Para tanto, uma placa auxiliar foi confeccionada com um circuito integrado MAX 232. Esta placa, através do componente MAX 232, converte o sinal de nível de tensão TTL para EIA 232. Assim, os dados adquiridos no barramento DeviceNet são enviados diretamente para o *buffer* da porta serial do computador. Na figura 16 está uma imagem mais detalhada da placa, onde pode-se verificar o fluxo de dados. Os dados trafegando no barramento DeviceNet são capturados e tratados pelo kit. A comunicação com o kit (gravação do programa e depuração) ocorre pela USB 1. O envio dos dados tratados ao computador ocorre pela USB 2.

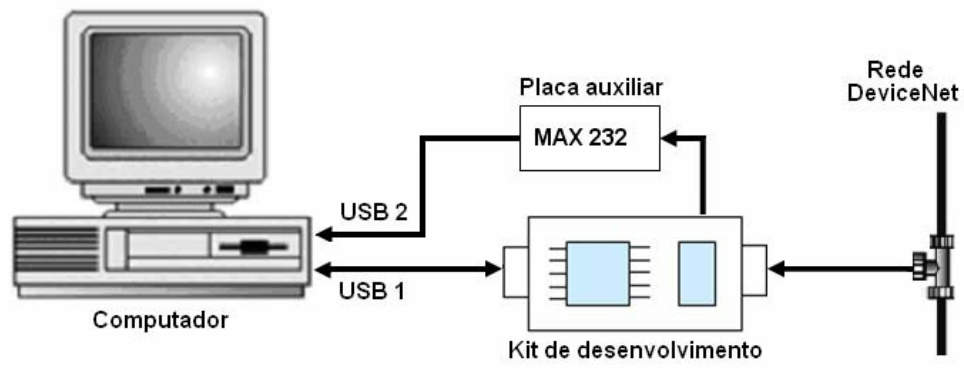


Figura 16 Sistema de monitoração.

A placa auxiliar foi conectada aos pinos de comunicação serial do microcontrolador, capturando os dados diretamente das trilhas do circuito impresso do kit de desenvolvimento. Com isso, os dados são capturados e enviados para outra porta USB sem comprometer o funcionamento do kit.

6.2 Software

A solução em software encontrada para implementar o sistema em questão pode ser separada em duas partes: software embarcado no microcontrolador e software de interface com o usuário desenvolvido no computador. O software para o microcontrolador foi desenvolvido em linguagem de programação C e faz a aquisição dos dados do barramento, armazenando as mensagens cíclico e enviando para sua porta de comunicação serial. Já o programa para o computador foi desenvolvido em linguagem de programação C++, que recebe os dados enviados pelo microcontrolador, armazenando os mesmos em um vetor para

posterior tratamento e exibição. A exibição ocorre na forma de listagem das mensagens, conforme sua ordem de captura no barramento.

Acompanhando o kit de desenvolvimento, vem um ambiente de compilação completo para programação do microcontrolador, o Hitop da empresa Hitex Development Tools GmbH. Este ambiente foi utilizado na implementação do trabalho proposto. O compilador traz inclusos alguns programas de exemplo, que foram adaptados e executados no kit. Dentre os arquivos inclusos, destacam-se códigos de demonstração para controle de temporizadores, interface serial e comunicação com o barramento CAN.

A partir de tais exemplos, fez-se um estudo e análise das partes do código referentes à comunicação com o barramento CAN. Identificou-se os trechos que determinavam a taxa de transmissão de dados, tamanho de *buffer* de armazenamento, comunicação com o barramento CAN e comunicação com o canal serial.

Alguns desses parâmetros foram alterados, como a taxa de transmissão, que originalmente estava definida em 100 kbit/s. Esta limitação constituía um problema, impossibilitando seu uso no presente trabalho, visto que a menor taxa DeviceNet é 125 kbit/s. Assim sendo, códigos foram alterados para operar em taxas de 125 kbit/s.

Na arquitetura da proposta, o módulo de monitoração DeviceNet não possui exibição em tempo-real. Assim, a taxa de aquisição do barramento DeviceNet é totalmente independente da interface de comunicação com o computador. O envio dos dados para o computador é feito através da porta USB, mas com a velocidade máxima limitada em 115200 bit/s, já que esse é o máximo que o canal serial do microcontrolador pode transmitir. A leitura do barramento DeviceNet, para os estudos de casos montados, ocorreram a uma taxa de 125 kbit/s. A exibição dos dados ocorre posteriormente.

Na figura 17 está uma visão geral de como deve operar a placa de interface entre o barramento CAN e o computador. O programa gravado no microcontrolador faz o controle dos dados adquiridos no barramento, identificando as mensagens DeviceNet. No instante em que cada mensagem é adquirida, também é armazenado o momento da aquisição, sendo ambos alocados no *buffer* de mensagens do microcontrolador.

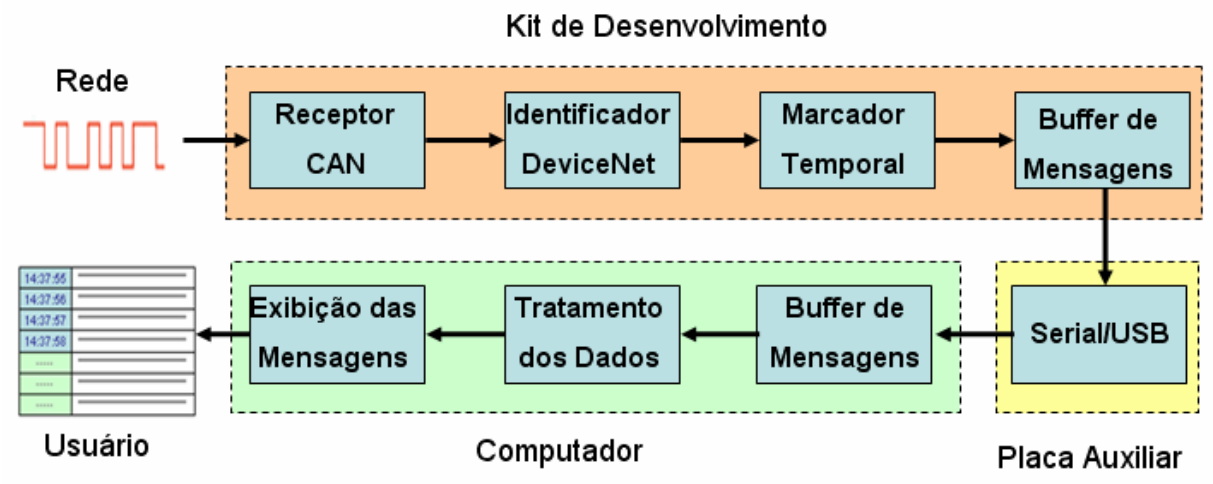


Figura 17 Diagrama de blocos do monitorador DeviceNet.

O *buffer* de mensagens utilizado no microcontrolador possui um tamanho definido de quatro mensagens. Apesar dos dados trafegarem a 125 kbit/s na rede DeviceNet, como há espaços de tempo entre as mensagens DeviceNet, um buffer de 4 mensagens guarda os dados pelo tempo suficiente para enviar pela porta serial antes dos mesmos serem sobrescritos. Com isso, fazem-se aquisições de dados na velocidade de operação do barramento, enviando os mesmos para o computador conforme a velocidade da serial.

Além do compilador, também acompanha o kit um software para geração e aquisição de mensagens CAN, o software chamado “U CAN Smart View”, da própria Infineon. O Smart View é um aplicativo gráfico que permite a geração e aquisição de mensagens CAN. Em conjunto com este aplicativo são fornecidos arquivos fonte demonstrativos. Tal software foi usado durante a etapa de desenvolvimento do projeto para auxiliar na implementação do mesmo, servindo como referência para conferência dos dados adquiridos.

A partir desta base, foi desenvolvido um software próprio para o computador, o qual exibe as mensagens que trafegam no barramento DeviceNet, especificando o grupo de mensagem, identificador MAC, campo de dados e comprimento do mesmo, assim como o momento da aquisição das mensagens e seu significado.

O software para o computador foi desenvolvido em linguagem C++, com o compilador Microsoft Visual C++ 2008 Express Edition. A partir do mesmo, pôde-se adquirir os dados que o programa do microcontrolador envia pela serial e fazer o adequado tratamento dos mesmos. Ao clicar-se na opção “Iniciar Aquisição”, os dados são armazenados em um vetor de tamanho variável de forma seqüencial conforme o microcontrolador vai enviando. Ao final da aquisição (preenchimento do vetor) inicia-se o tratamento dos dados. O vetor com

os dados adquiridos é percorrido em busca do byte que indique o início de uma mensagem. A partir disso são feitos vários testes para confirmar que se trata realmente de uma mensagem DeviceNet. Confirmando-se o início da mensagem, são identificados os campos do identificador da mensagem. Com isso, pode-se determinar o grupo da mensagem, o identificador MAC e as informações de origem ou destino. Após o identificador, encontra-se a informação do comprimento do campo de dados, seguido pelos dados propriamente ditos. Ao final, consta o instante em que a mensagem acabou.

Uma visão geral do software desenvolvido está na figura 18. A partir dos dados enviados pelo microcontrolador, o software deverá exibir as mensagens na tela. Na exibição, deve constar o momento da aquisição de cada mensagem e também o conteúdo do cabeçalho da mensagem, especificando dados como o grupo de mensagem, identificador MAC e identificador de mensagem. O módulo de interface com o usuário aceita comandos como inicio e fim da exibição.

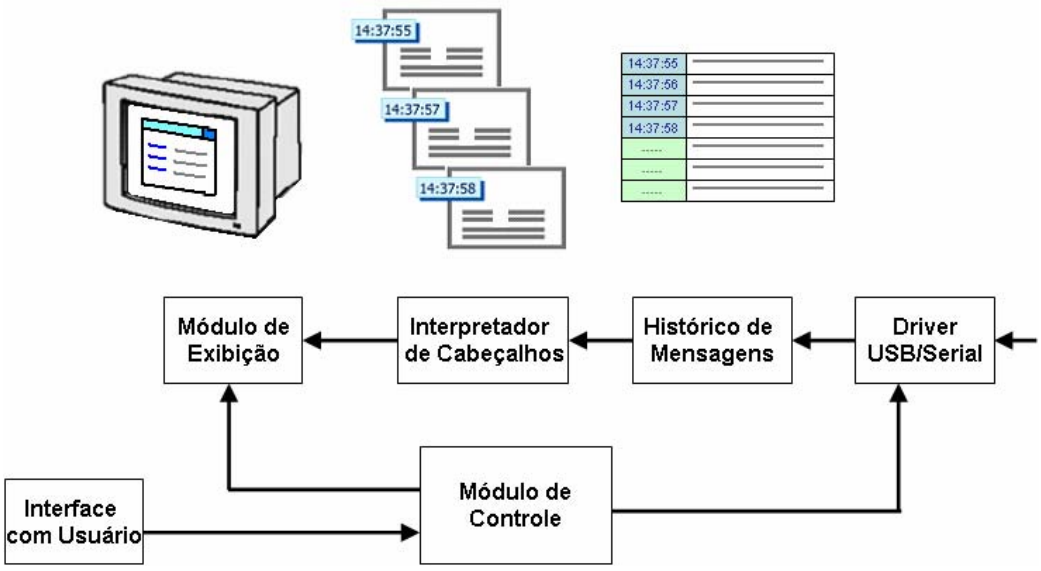


Figura 18 Diagrama de blocos do software desenvolvido.

Na figura 19 é apresentada uma imagem com detalhe da tela do software de exibição desenvolvido. Por ser uma versão didática, alguns dados presentes são apenas para fins de conferência durante a execução do programa.

Figura 19 Detalhe da tela de exibição do software desenvolvido.

Dentre os campos que podem ser vistos, estão as informações citadas como sendo o objeto de estudo do presente trabalho, na seqüência:

- **Tempo** – Indica o tempo em micro segundos entre duas mensagens. Com isso, sabendo-se a velocidade que a rede está configurada e o tamanho da mensagem enviada, pode-se determinar o intervalo de tempo que o barramento ficou ocupado e ocioso. Se faz necessário destacar que a indicação de tempo da primeira mensagem deve ser desconsiderada, pois não há mensagem anterior para fins de comparação.
- **Identificador** – Apresenta o campo do identificador de mensagem CAN, que é a junção do valor do identificador MAC com o identificador de cada mensagem do grupo. É por meio deste que pode-se chegar ao significado da mensagem determinado pelas especificações DeviceNet.
- **Grupo** – Indica a qual grupo de mensagens pertence a mensagem. Dependendo do valor de identificador, há um grupo de mensagens correspondente, que segundo as especificações pode ser o grupo 1, 2, 3 ou 4.
- **MAC ID** – Neste espaço é apresentado o identificador MAC do dispositivo em questão. Dependendo do tipo de mensagem, pode ser tanto o MAC do dispositivo de origem quanto do dispositivo de destino. Quando existir, seu valor será obrigatoriamente no máximo 63.
- **Tamanho** – Indica qual o comprimento do campo de dados, ou seja, qual a quantidade de bytes presentes no campo de dados da mensagem.
- **Dados** – Apresenta os dados que a mensagem está carregando em seu campo específico. Os valores presentes podem tanto ser dados de I/O trocados entre os

dispositivos, como também informações de configuração do sistema, dependendo do tipo de mensagem.

- **Significado** – Neste espaço são apresentados os significados das mensagens que trafegam no barramento. Cada conjunto de identificadores apresenta um determinado significado. Durante a implementação, optou-se por não traduzir este campo para língua portuguesa a fim de não prejudicar seu significado.

7 PARTE EXPERIMENTAL

A fim de validar o sistema desenvolvido, este foi testado em uma rede DeviceNet real, montada e configurada no Laboratório de Automação Industrial da Univates. Para os experimentos, foram utilizados dois CLPs da fabricante Rockwell Automation, modelo “CompactLogix 1769” disponíveis no laboratório. Cada CLP possui uma CPU modelo L32E e módulos de expansão de entradas e saídas digitais e analógicas, além de um módulo para comunicação em barramento DeviceNet, o Scanner DeviceNet 1769-SDN, cuja imagem pode ser vista na figura 20.

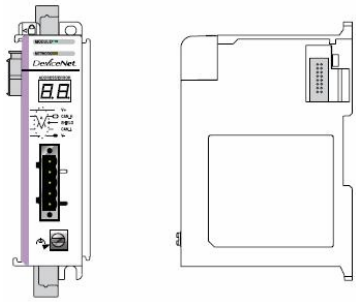


Figura 20 Vista frontal e lateral do módulo scanner.

Para realização dos ensaios, os dois CLPs foram conectados por um barramento DeviceNet através de seus módulos de comunicação 1769-SDN. Ao cabo de comunicação foi adicionada uma derivação, ligada ao protótipo desenvolvido (kit de desenvolvimento U CAN Start Kit). Os dados que trafegavam no barramento eram então capturados pelo kit, que se comunicava com o computador por uma porta USB. Para envio dos dados adquiridos foi utilizada uma placa auxiliar que convertia a saída serial do kit de desenvolvimento para a interface USB. Uma imagem do sistema montado é apresentada na figura 21.

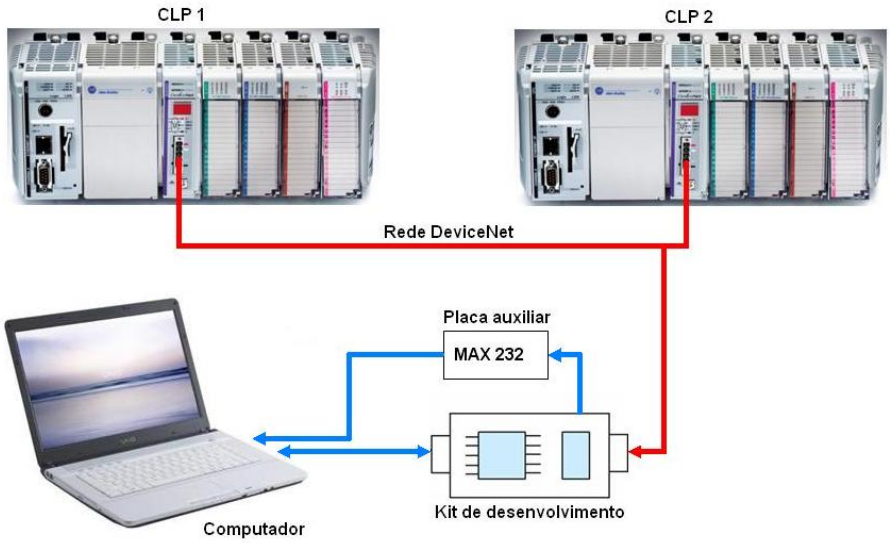


Figura 21 Sistema montado para realização dos testes.

Com esta estrutura montada, testes foram realizados em diferentes configurações de rede. Foram alterados parâmetros como os valores dos dados enviados, quantidade de dados, tipo de mensagens e identificador MAC dos dispositivos. Os resultados obtidos são apresentados nas próximas seções.

7.1 Ensaio 1

Para a configuração da rede implementada durante os testes, utilizou-se o software RSNetWorx for DeviceNet da empresa Rockwell Automation. Esse software é usado para configuração dos dispositivos em uma rede DeviceNet, que no presente caso é composta pelos dois CLPs. Após estabelecida a conexão com os CLPs, o programa faz uma varredura e detecta todos os dispositivos existentes na rede. A partir disso, pode-se alterar o identificador MAC dos CLPs. No ensaio realizado foram definidos com os valores 1 e 2, como pode ser visto na figura 22.

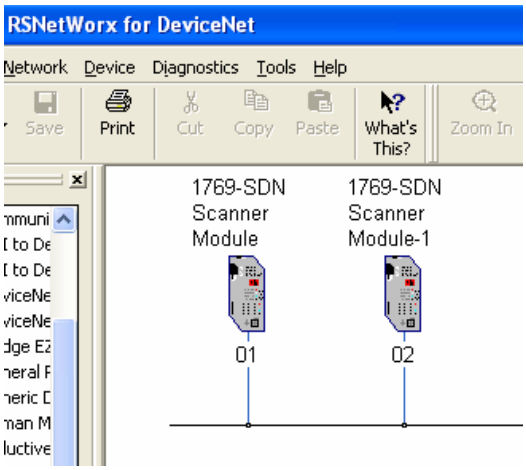


Figura 22 Detalhe da tela de configuração do RSNetWorx for DeviceNet.

Por definição do fabricante, os CLPs vêm configurados como mestres, então um deles teve que ser configurado para operar como escravo, permitindo assim a troca de dados entre os dois. Para tanto, o modo escravo de um dos CLPs foi habilitado. Neste caso, definiu-se que o CLP 2 seria o escravo, enquanto o CLP 1 atuaria como mestre. A figura 23 apresenta uma imagem da janela de configuração do modo escravo.

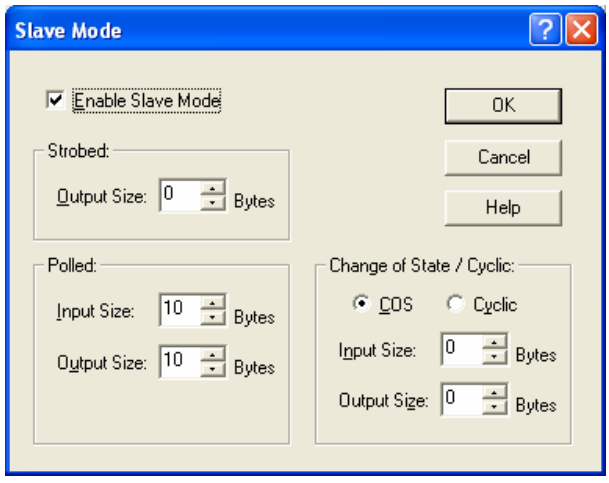


Figura 23 Aba de configuração do modo escravo.

Na aba *Slave Mode* também é definido o modo de operação que será usado para comunicação entre o mestre e o escravo. Habilitou-se o modo *Polled*, sendo destinados 10 bytes para entrada de dados e 10 bytes para saída. A velocidade de comunicação entre mestre e escravo foi definida em 125kbit/s. Com um CLP já definido como escravo e a sua forma de envio e tamanho das mensagens, ele pode agora ser adicionado ao *Scanlist* do CLP mestre.

Os 10 bytes de entrada e saída do escravo devem ser também mapeados na memória do mestre. A partir desse mapeamento sabe-se então quais os endereços que devem ser usados

para escrita e leitura dos dados no barramento, tanto no mestre quanto no escravo. Como no presente caso havia apenas um mestre e um escravo na rede, as posições mapeadas para ambos foram as mesmas. Tal procedimento pode ser visto na figura 24.

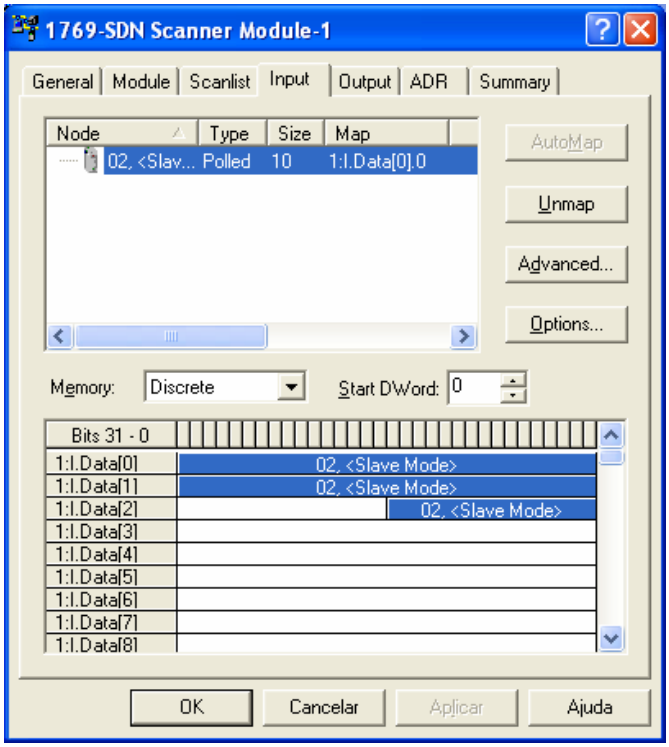


Figura 24 Tela de mapeamento das posições de memória.

Após isso, pode-se utilizar os endereços correspondentes aos dados citados no software de programação dos CLPs para que a troca de dados através da rede DeviceNet seja efetuada.

Para o ensaio realizado, foi definida uma troca de dados de variáveis digitais, simulando entradas digitais que poderiam ser dispositivos discretos externos como, por exemplo, chaves fim de curso. A título de demonstração, um conjunto de chaves (*dipswitch)* de oito posições foi colocada nas oito primeiras entradas do módulo de entradas digitas do CLP 2 (escravo). Essas oito entradas foram mapeadas nas oito primeiras posições de memória reservadas para comunicação DeviceNet. Tais posições seriam então lidas pelo CLP 1 (mestre), colocando os valores diretamente nas oito primeiras posições do módulo de saída digital. A programação dos CLPs foi feita com o software RSLogix 5000 também da Rockwell Automation. A linguagem de programação Ladder foi utilizada para tal implementação. Na figura 25 está uma representação do sistema.

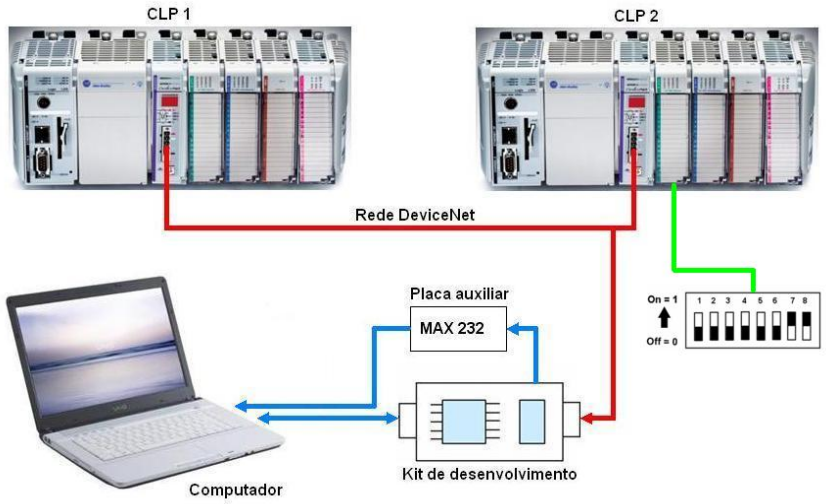


Figura 25 Sistema montado utilizando uma *dipswitch*.

Foram feitas mudanças nas posições das chaves da *dipswitch* e uma captura dos dados para cada mudança, conforme as imagens capturadas do software de monitoração (figura 27). Quatro combinações de posições da *dipswitch* foram feitas e os dados do barramento capturados. Conforme as combinações mudam, pode-se perceber a alteração dos dados trafegando no barramento As combinações testadas foram:

a) todas as chaves em “0”, ou seja, todas as entradas digitais em nível lógico “0”;
b) as quatro chaves que representam os bits menos significativos (1...4) em nível lógico “1” e as de mais em “0”;
c) as quatro chaves que representam os bits mais significativos (5...8) em nível lógico “1” e as demais em “0”;
d) todas as oito chaves em nível lógico “1”.

Na figura 26 está uma imagem da tela do software de monitoração após a captura das mensagens na condição em que a *dipswitch* estava com todas as chaves na posição “0”.

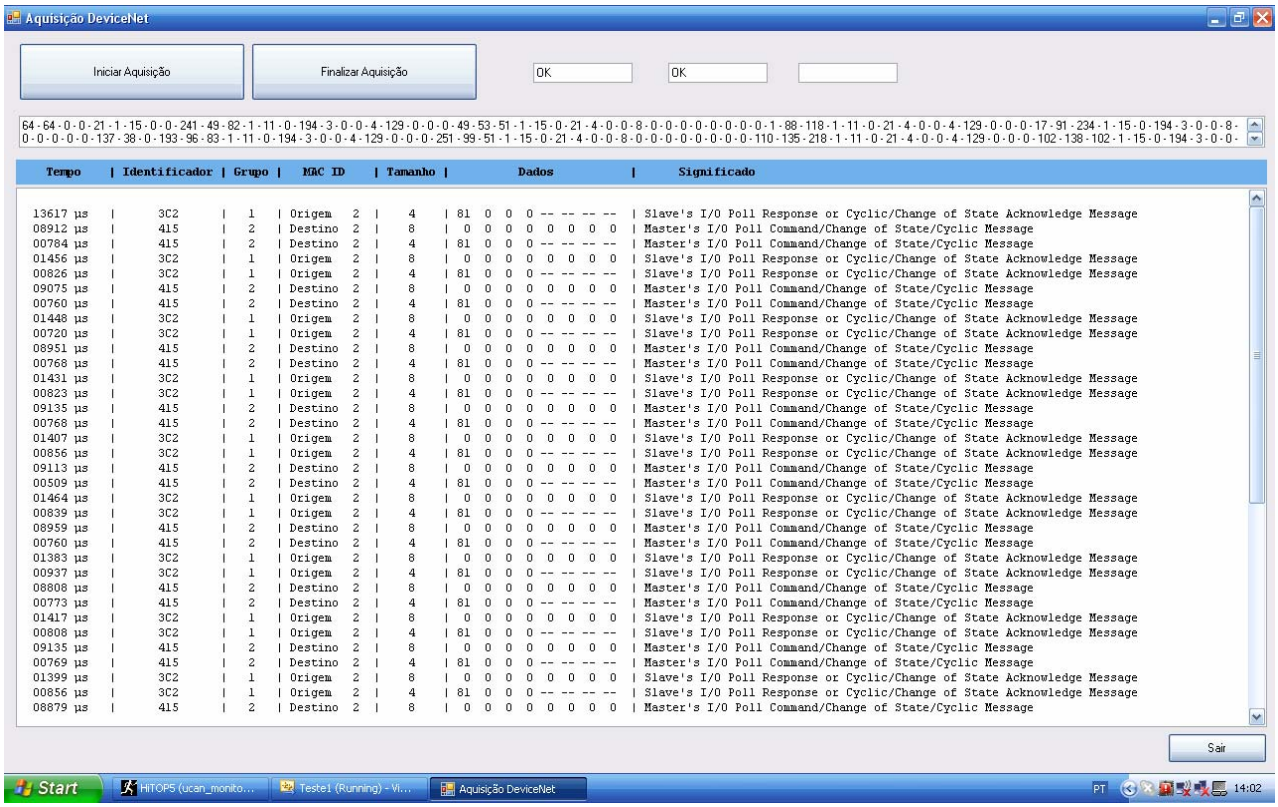


Figura 26 Captura da tela do software desenvolvido com *dipswitch* em "0".

Pode-se observar que as mensagens se repetem na tela em períodos praticamente iguais, sendo duas mensagens do mestre e duas mensagens do escravo. Conforme foi programado no software de tratamento e exibição dos dados, as mensagens do mestre pertencem ao grupo 2 e têm como destino o dispositivo com identificador MAC = 2, ou seja, o CLP que foi configurado como escravo e está na posição 2 do barramento. Como foi definido no software de configuração de rede RSNetWorx, as mensagens trocadas entre os dois dispositivos seriam apenas mensagens *poll*, ou seja, mensagens de pesquisa. É o que pode ser visto no campo "Significado" apresentado pelo software desenvolvido. As mensagens do mestre possuem o mesmo identificador e indicam: "Master's I/O Poll Command". Pode-se observar na mensagem do mestre que o identificador MAC carrega o valor do escravo de destino.

O campo chamado tamanho indica o número de bytes existentes no campo de dados. O fato de reservar-se 10 bytes de entrada e saída na memória dos dois CLPs causa a necessidade do envio de mensagens fragmentadas. Como o número máximo de bytes que uma mensagem DeviceNet pode transportar é 8 bytes, são necessárias duas mensagens para transportar todos os 10 bytes de dados.

Em mensagens fragmentadas, o primeiro byte de cada mensagem é usado para informações do sistema. Os sete bytes seguintes contém então os dados que realmente devem ser enviados (ODVA, 2007d).

As mensagens do escravo também estão corretas, pois apresentam o valor do MAC ID do dispositivo de origem (no caso o CLP 2). Observa-se adicionalmente que o campo de dados do escravo vai sendo alterado conforme mudam-se as posições das chaves, como visto na figura 27.

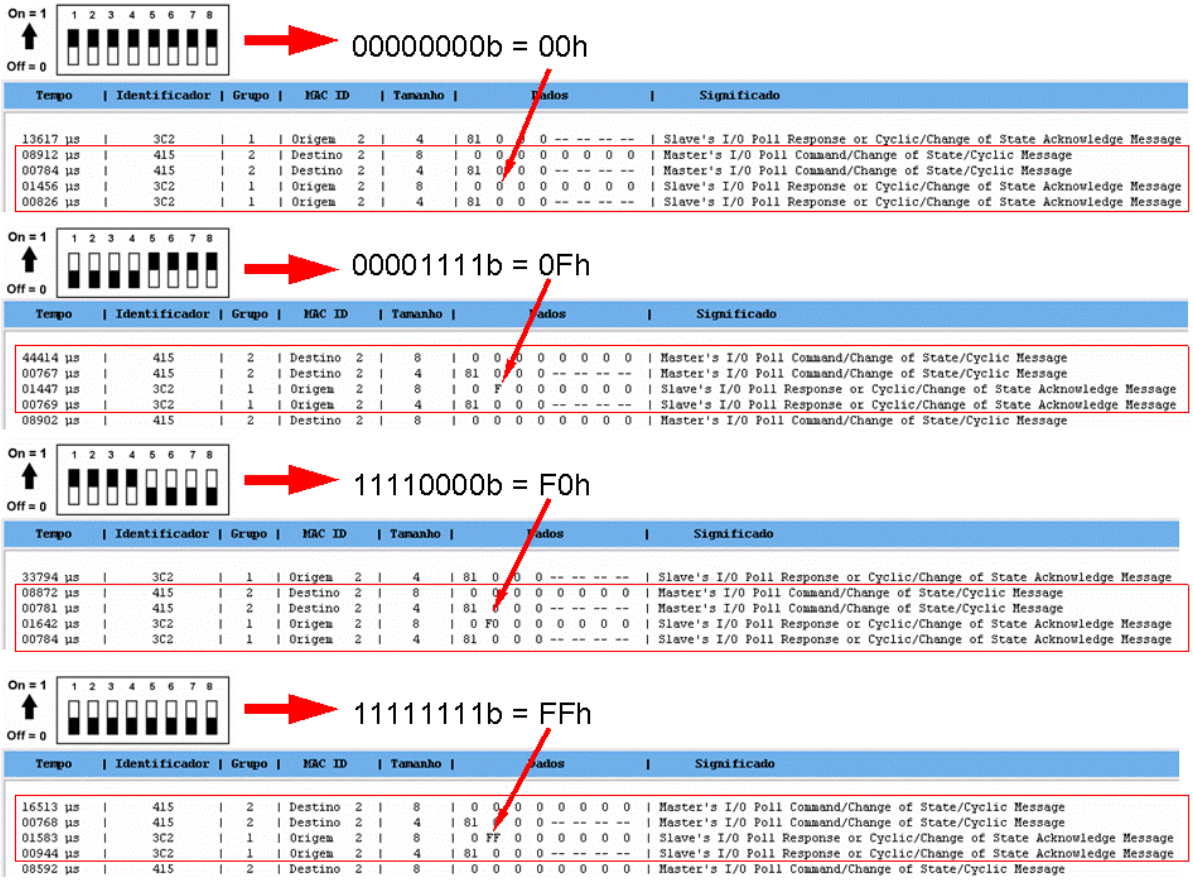


Figura 27 Detalhe da tela do software para as variações na *dipswitch*.

O recurso de fragmentação usado pelo barramento DeviceNet consegue ser facilmente identificado a partir da ferramenta proposta. Particularmente observa-se que os valores do campo de dados possuem seguinte significado:

Mestre (primeira e segunda mensagem)

00 → tipo de fragmento: 0 = fragmento inicial/ 0 = contagem de fragmentos.

Sete bytes seguintes = 0 → todas as variáveis transmitidas possuem valor “0”.

81 → tipo de fragmento: 8 = fragmento final/ 1 = contagem de fragmento.

Escravo (terceira e quarta mensagem)

00 → tipo de fragmento: 0 = fragmento inicial/ 0 = contagem de fragmentos.

NN → valor do dado transmitido = combinações da *dipswitch.*

Seis bytes seguintes = 0 → as variáveis seguintes possuem valor “0”.

81 → tipo de fragmento: 8 = fragmento final/ 1 = contagem de fragmentos.

Três bytes seguintes = 0: todas as variáveis transmitidas possuem valor “0”.

Uma representação em destaque deste recurso de fragmentação é apresentada na figura 28, onde se identificam as posições na mensagem dos 10 bytes de dados enviado pelo escravo. O primeiro byte de cada fragmento é reservado para envio de informações sobre a fragmentação. Os bytes seguintes são então preenchidos com os dados (somente 7 bytes de dados são transmitidos em cada mensagem). Por esse motivos há 8 bytes na primeira mensagem (1 byte de informação + 7 bytes de dados) e 4 bytes na segunda mensagem (1 byte de informação + 3 bytes de dados).

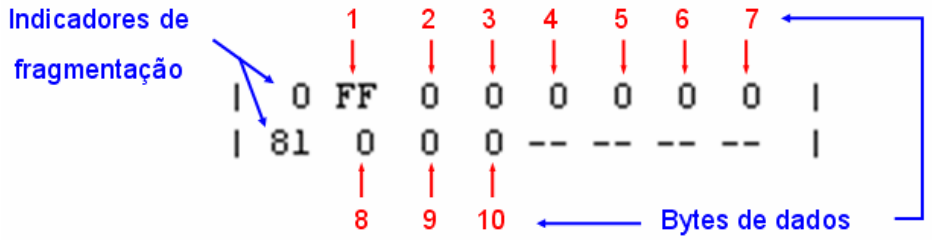


Figura 28 Detalhe do campo de dados na mensagem fragmentada.

Este ensaio serve para demonstrar a importância de uma rede bem implementada. No exemplo, foram reservados 10 bytes para troca de dados, mas na verdade somente 1 deles estavam sendo utilizado. Com isso, uma mensagem adicional estava trafegando de forma desnecessária no barramento. Apesar de os bytes não estarem todos sendo usados pelo software de programação dos CLPs, o fato de serem reservados no momento da configuração da rede já provoca o seu envio.

7.2 Ensaio 2

Para o segundo ensaio determinou-se o envio de uma variável coletada através do módulo de entradas analógicas do CLP escravo. A configuração do barramento foi mantida a mesma do ensaio anterior, ou seja, mensagens tipo *poll* com 10 bytes de entrada e 10 bytes de saída. Os identificadores MAC também foram mantidos os mesmos (mestre com valor “1” e escravo com valor “2”).

Foi prevista uma largura de 16 bits para as entradas analógicas dos CLPs, apesar de na prática as conversões A/D (Analógico/Digital) do CLP utilizado possuírem resolução de 14 bits. Com isso, reservou-se 2 bytes para guardar o valor da variável analógica e enviá-lo pelo barramento DeviceNet. Configurou-se estes 16 bits para serem armazenados na parte menos significativa do segundo conjunto de bytes na memória dos CLPs. Na figura 29 pode-se ver uma imagem do sistema montado com os dois CLPs e a fonte de tensão regulável ligada ao CLP escravo.

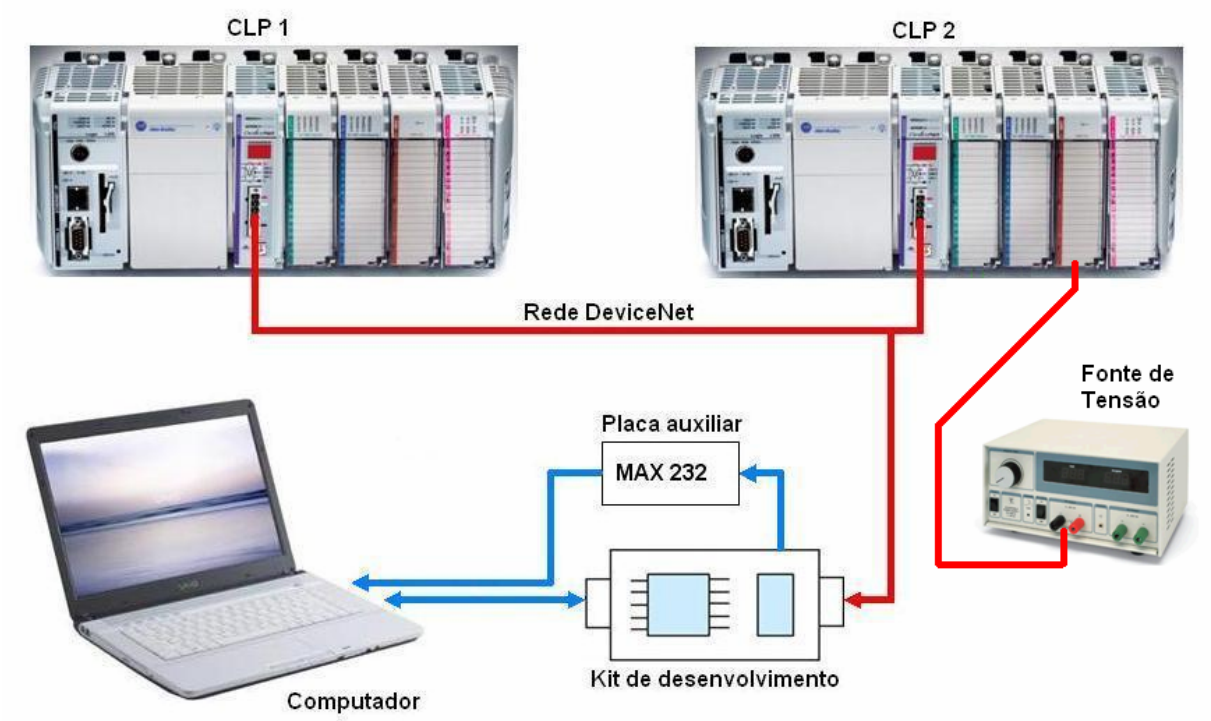


Figura 29 Sistema montado utilizando uma fonte de tensão regulável.

A partir do sistema montado e configurado, variou-se o valor da fonte de tensão em diferentes aquisições de dados do barramento. Os valores reais capturados são mostrados juntamente com imagens do software de programação dos CLPs, o RSLogix 5000.

Na figura 30 pode-se ver imagens do software capturadas com a fonte ajustada para valores em torno de 0 V, 5 V e 10 V. Os valores lidos pelo CLP são ligeiramente diferentes dos valores indicados no display da fonte, devido à imprecisão na resolução da tensão medida. Este aspecto entretanto não altera o propósito do teste, que era o de monitorar a troca de dados entre os CLPs.

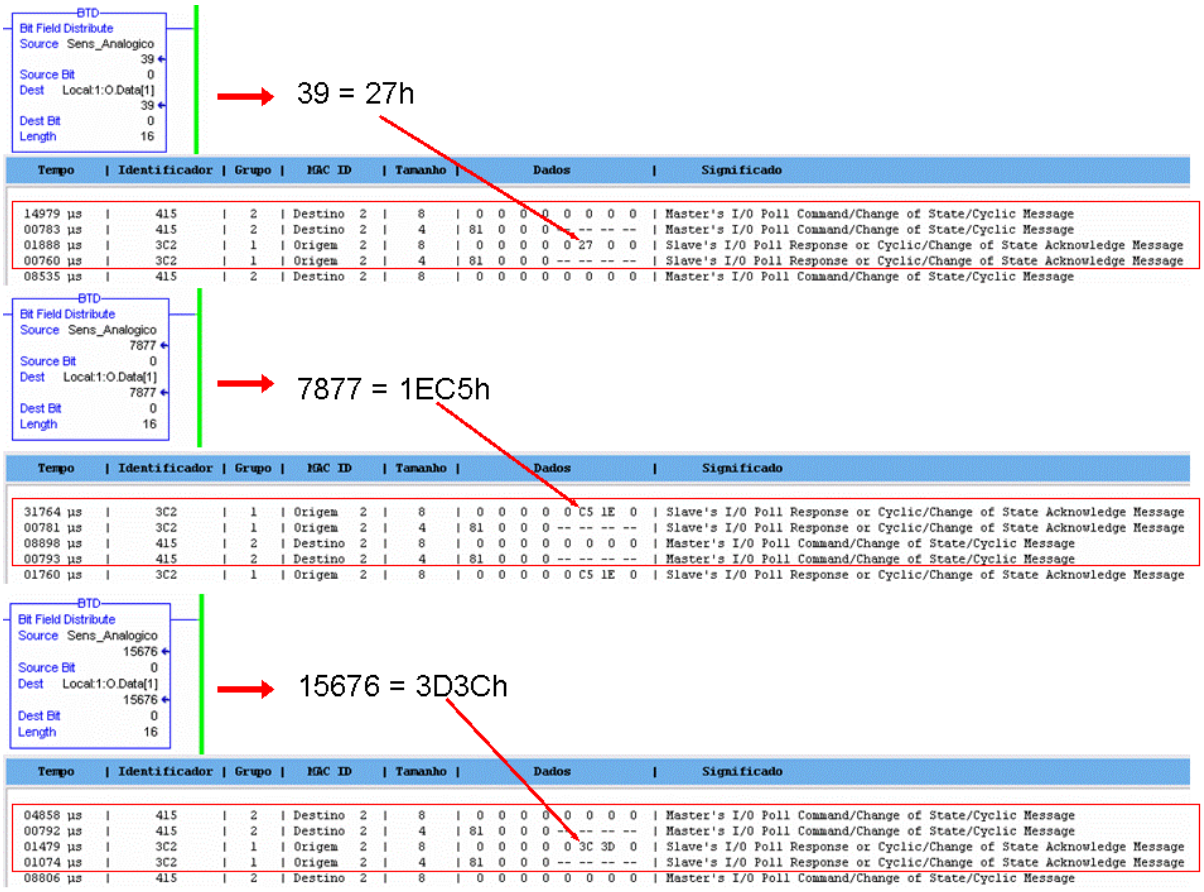


Figura 30 Detalhe da tela do software para as variações na fonte de tensão.

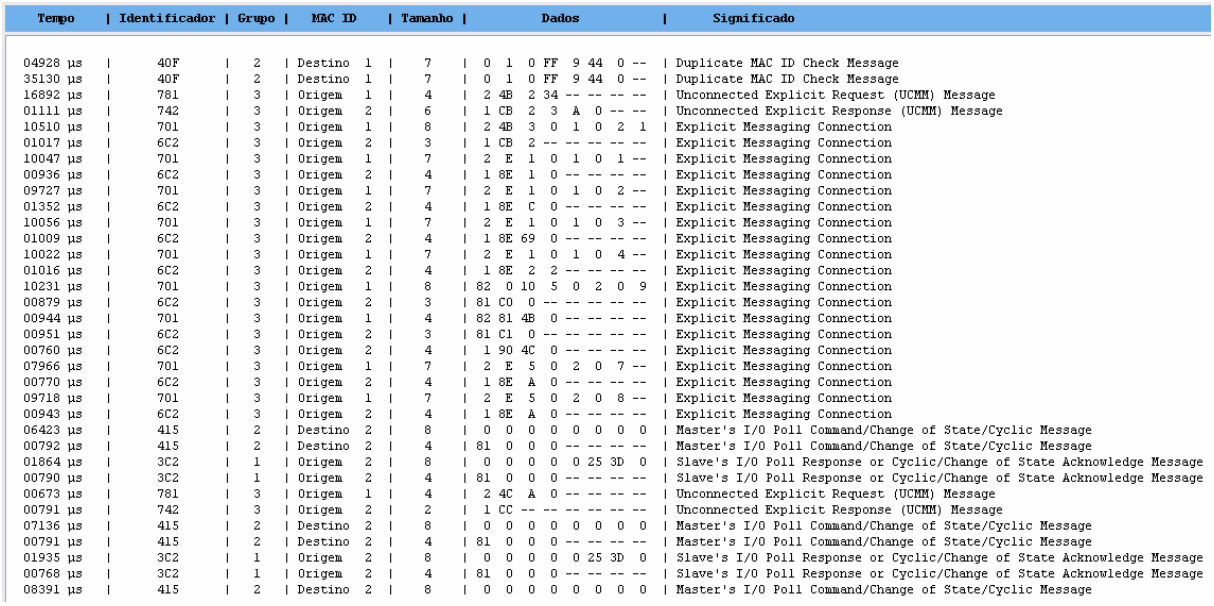
Da mesma forma que ocorreu na implementação anterior, o fato de ter-se reservado 10 bytes de entrada e 10 bytes de saída para os CLPs, gerou a fragmentação das mensagens. Conforme definido, os 2 bytes com o valor analógico ocupam os 2 bytes menos significativos da segunda parte da área de memória reservada, (posições 5 e 6 dos dados). Para cada um dos testes realizados variando-se a voltagem da fonte, pode-se perceber na figura 30 que os valores na entrada do CLP aparecem sendo transmitidos no espaço reservado.

7.3 Ensaio 3

O terceiro ensaio foi realizado com o intuito de demonstrar a transferência de dados entre os CLPs durante a entrada de um dispositivo no barramento. Para tanto, a configuração foi mantida idêntica à dos testes iniciais, ou seja, a quantidade de dados de entrada e saída foi definida em 10 bytes, tipo de mensagem *poll* e os identificadores MAC com valores “1” para o mestre e “2” para o escravo.

Com estas configurações implementadas, iniciou-se a monitoração do barramento apenas com o escravo conectado. Neste caso não há tráfego algum de dados. Então, conectou-

se o mestre ao barramento e capturou-se todo o processo de identificação e estabelecimento de troca de mensagens entre estações. Estes resultados podem ser vistos na figura 31.



Tempo	Identificador	Grupo	MAC ID	Tamanho	Dados	Significado
04928 µs	40F	2	Destino 1	7	0 1 0 FF 9 44 0 --	Duplicate MAC ID Check Message
35130 µs	40F	2	Destino 1	7	0 1 0 FF 9 44 0 --	Duplicate MAC ID Check Message
16892 µs	781	3	Origem 1	4	2 4B 2 34 -- -- -- --	Unconnected Explicit Request (UCMM) Message
01111 µs	742	3	Origem 2	6	1 CB 2 3 A 0 -- --	Unconnected Explicit Response (UCMM) Message
10510 µs	701	3	Origem 1	8	2 4B 3 0 1 0 2 1	Explicit Messaging Connection
01017 µs	6C2	3	Origem 2	3	1 CB 2 -- -- -- -- --	Explicit Messaging Connection
10047 µs	701	3	Origem 1	7	2 E 1 0 1 0 1 --	Explicit Messaging Connection
00936 µs	6C2	3	Origem 2	4	1 8E 1 0 -- -- -- --	Explicit Messaging Connection
09727 µs	701	3	Origem 1	7	2 E 1 0 1 0 2 --	Explicit Messaging Connection
01352 µs	6C2	3	Origem 2	4	1 8E C 0 -- -- -- --	Explicit Messaging Connection
10056 µs	701	3	Origem 1	7	2 E 1 0 1 0 3 --	Explicit Messaging Connection
01009 µs	6C2	3	Origem 2	4	1 8E 69 0 -- -- -- --	Explicit Messaging Connection
10022 µs	701	3	Origem 1	7	2 E 1 0 1 0 4 --	Explicit Messaging Connection
01016 µs	6C2	3	Origem 2	4	1 8E 2 2 -- -- -- --	Explicit Messaging Connection
10231 µs	701	3	Origem 1	8	82 0 10 5 0 2 0 9	Explicit Messaging Connection
00879 µs	6C2	3	Origem 2	3	81 C0 0 -- -- -- -- --	Explicit Messaging Connection
00944 µs	701	3	Origem 1	4	82 81 4B 0 -- -- -- --	Explicit Messaging Connection
00951 µs	6C2	3	Origem 2	3	81 C1 0 -- -- -- -- --	Explicit Messaging Connection
00760 µs	6C2	3	Origem 2	4	1 90 4C 0 -- -- -- --	Explicit Messaging Connection
07966 µs	701	3	Origem 1	7	2 E 5 0 2 0 7 --	Explicit Messaging Connection
00770 µs	6C2	3	Origem 2	4	1 8E A 0 -- -- -- --	Explicit Messaging Connection
09718 µs	701	3	Origem 1	7	2 E 5 0 2 0 8 --	Explicit Messaging Connection
00943 µs	6C2	3	Origem 2	4	1 8E A 0 -- -- -- --	Explicit Messaging Connection
06423 µs	415	2	Destino 2	8	0 0 0 0 0 0 0 0	Master's I/O Poll Command/Change of State/Cyclic Message
00792 µs	415	2	Destino 2	4	81 0 0 0 -- -- -- --	Master's I/O Poll Command/Change of State/Cyclic Message
01864 µs	3C2	1	Origem 2	8	0 0 0 0 0 25 3D 0	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
00790 µs	3C2	1	Origem 2	4	81 0 0 0 -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
00673 µs	781	3	Origem 1	4	2 4C A 0 -- -- -- --	Unconnected Explicit Request (UCMM) Message
00791 µs	742	3	Origem 2	2	1 CC -- -- -- -- -- --	Unconnected Explicit Response (UCMM) Message
07136 µs	415	2	Destino 2	8	0 0 0 0 0 0 0 0	Master's I/O Poll Command/Change of State/Cyclic Message
00791 µs	415	2	Destino 2	4	81 0 0 0 -- -- -- --	Master's I/O Poll Command/Change of State/Cyclic Message
01935 µs	3C2	1	Origem 2	8	0 0 0 0 0 25 3D 0	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
00768 µs	3C2	1	Origem 2	4	81 0 0 0 -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
08391 µs	415	2	Destino 2	8	0 0 0 0 0 0 0 0	Master's I/O Poll Command/Change of State/Cyclic Message

Figura 31 Tela do software no momento da entrada do dispositivo.

Pode-se observar que primeiramente o escravo envia, por definição, duas mensagens de verificação de identificador MAC duplicado, que neste caso não está duplicado. Em seguida, o mestre envia uma mensagem requisitando o estabelecimento de uma conexão de mensagens explícitas, definido pelo byte com valor 4Bh. Como resposta, o CLP escravo retorna uma mensagem confirmando tal pedido, definido no byte CBh (ODVA, 2007d).

Em seguida, são transmitidas algumas mensagens entre o mestre e o escravo definidas como conexão de mensagens explícitas, através das quais são trocados parâmetros de serviços específicos de configuração. Depois delas, são transmitidas uma única vez as mensagens que irão trafegar no barramento.

Com esta etapa concluída, o CLP mestre envia uma mensagem requisitando o fechamento da conexão de mensagens explícitas, definido pelo byte com valor 4Ch, que é respondido pelo escravo com uma mensagem de confirmação, definido pelo byte CCh. Deste ponto em diante, as mensagens que trafegam no barramento são as mesmas demonstradas na seção 7.2.

7.4 Ensaio 4

No quarto experimento, alguns parâmetros foram alterados na configuração do barramento, a fim de se observar a mudança no tráfego de dados. Neste caso a quantidade de dados reservados para entrada e saída dos CLPs foi alterada para 8 bytes cada. Outro parâmetro alterado foi o identificador MAC do CLP escravo, que antes possuía o valor 2 e agora possui o valor 3. Tais mudanças podem ser vistas nas figuras 32 e 33, onde aparecem os valores do identificador MAC dos CLPs e a configuração do modo escravo, respectivamente.

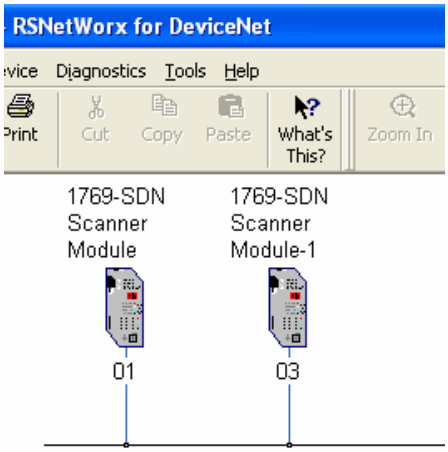


Figura 32 Detalhe da tela de configuração do RSNetWorx for DeviceNet.

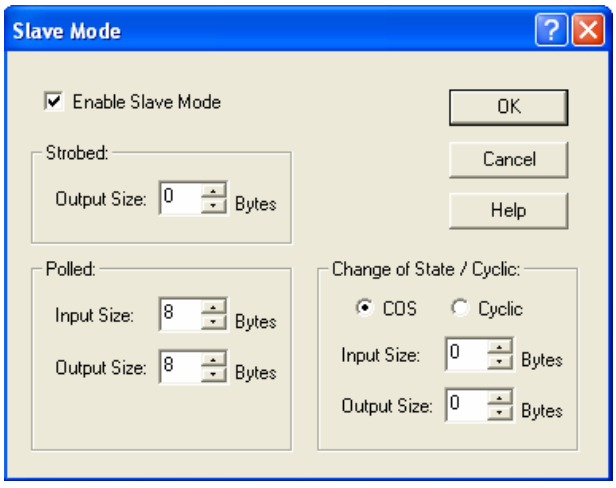
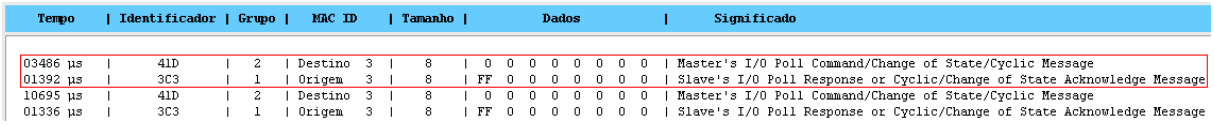


Figura 33 Aba de configuração do modo escravo.

A partir desta implementação, realizou-se a captura dos dados trafegados no barramento. Para fins de simulação de entradas digitais, definiu-se no próprio software de programação dos CLPs que os 8 bits menos significativos ficariam em nível lógico “1”.

Com a definição de apenas 8 bytes de entrada e saída, as mensagens no barramento não precisam ser fragmentadas. Isso pode ser observado na figura 34, onde todos os 8 bytes do campo de dados da mensagem são ocupados para o transporte de valores de variáveis.



Tempo	Identificador	Grupo	MAC ID	Tamanho	Dados	Significado
03486 µs	41D	2	Destino 3	8	0 0 0 0 0 0 0 0	Master's I/O Poll Command/Change of State/Cyclic Message
01392 µs	3C3	1	Origem 3	8	FF 0 0 0 0 0 0 0	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
10695 µs	41D	2	Destino 3	8	0 0 0 0 0 0 0 0	Master's I/O Poll Command/Change of State/Cyclic Message
01336 µs	3C3	1	Origem 3	8	FF 0 0 0 0 0 0 0	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message

Figura 34 Detalhe da tela do software com campo de dados de 8 bytes.

7.5 Ensaio 5

Neste ensaio, manteve-se a mesma configuração da rede, com exceção do tipo de mensagem enviada (CLP mestre e CLP escravo com os mesmos valores de identificadores MAC e quantidade de bytes para entrada e saída mantida em 8). O tipo de mensagem usada para troca de dados entre os dispositivos foi configurado para utilizar o mecanismo *Change of State* (COS) ao invés de *poll*. Tal configuração pode ser vista na figura 35, onde o CLP escravo passa a utilizar mensagens COS na troca de dados.

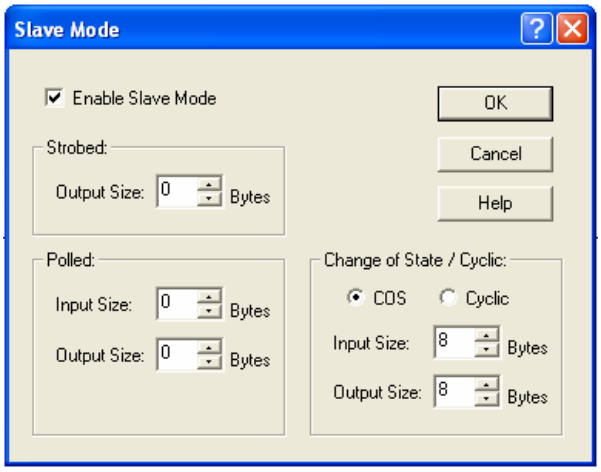


Figura 35 Aba de configuração do modo escravo.

Como resultado desta configuração, a aquisição de dados por parte do software monitorador apresenta um novo comportamento do barramento, como visto na figura 36.

Tempo	Identificador	Grupo	MAC ID	Tamanho	Dados	Significado
21518 µs	343	1	Origem 3	8	FF FF FF FF FF FF FF FF	Slave's I/O Change of State or Cyclic Message
00679 µs	41A	2	Destino 3	0	-- -- -- -- -- -- -- --	Master's Change of State/Cyclic Acknowledge Message
60090 µs	41D	2	Destino 3	8	FF FF FF FF FF FF FF FF	Master's I/O Poll Command/Change of State/Cyclic Message
00521 µs	3C3	1	Origem 3	0	-- -- -- -- -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
59856 µs	343	1	Origem 3	8	FF FF FF FF FF FF FF FF	Slave's I/O Change of State or Cyclic Message
00408 µs	41A	2	Destino 3	0	-- -- -- -- -- -- -- --	Master's Change of State/Cyclic Acknowledge Message
60106 µs	41D	2	Destino 3	8	FF FF FF FF FF FF FF FF	Master's I/O Poll Command/Change of State/Cyclic Message
00417 µs	3C3	1	Origem 3	0	-- -- -- -- -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message

Figura 36 Detalhe da tela do software com mensagens COS.

No campo de dados estão presentes valores definidos no software de configuração dos CLPs. Neste caso optou-se por colocar todos os bits do campo de memória reservado em nível lógico “1”.

Como conseqüência da alteração do tipo de mensagem utilizada, a seqüência de envio de mensagens se altera. Sempre que mensagens COS (identificadas no campo “Significado”) são enviadas por um dispositivo, é necessário que o dispositivo consumidor confirme o seu recebimento. Tal comportamento pode ser visto na figura 36. Pelo software de monitoração, percebe-se que quando o mestre disponibiliza seus dados no barramento (*Master’s I/O Change of State Message*) , em seguida o escravo retorna, por definição, uma mensagem com o campo de dados sem dados, confirmando o recebimento desta mensagem COS (*Slave’s Change of State Acknowledge Message*). Depois, o escravo envia uma mensagem, disponibilizando seus dados no barramento (*Slave’s I/O Change of State Message*). Com isso, logo em seguida o mestre envia uma mensagem, sem dados confirmando o recebimento da mesma (*Master’s Change of State Acknowledge Message*) (ODVA, 2007d).

Outro aspecto importante que pode ser observado na tela do monitorador é a significativa alteração no tempo de envio das mensagens. Ao contrário das mensagens *poll*, as mensagens COS devem ser enviadas somente no momento em que ocorre alteração no valor das variáveis relacionadas ao seu campo de dados. Caso não ocorra alterações dentro de um determinado intervalo de tempo, o dispositivo envia a mensagem para avisar que continua ativo no barramento, fato constatado com a ferramenta desenvolvida.

7.6 Ensaio 6

Neste ensaio, os valores dos identificadores MAC dos dois CLPs foram mantidos os mesmos, sendo alterados o tipo de mensagem (agora definido como *Cyclic*) e a quantidade de bytes de entrada e saída (reduzida para 6 bytes cada). Esta configuração pode ser vista na figura 37.

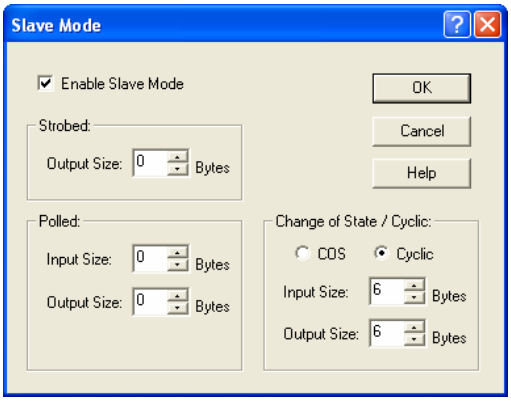


Figura 37 Aba de configuração do modo escravo.

Os dados trocados entre os CLPs também foram determinados no software de programação dos CLPs onde todos os bits seriam colocados em nível lógico “1”. Como pode ser visto na figura 38, o procedimento de envio de dados no barramento com mensagens *Cyclic* é basicamente o mesmo utilizado pelas mensagens *Change-of-State*, ou seja, sempre que o mestre envia seus dados, o escravo retorna uma mensagem confirmando o recebimento. O mesmo acontece quando o escravo envia uma mensagem de dados, fazendo o mestre retornar uma mensagem de confirmação. A principal diferença entre as mensagens COS e *Cyclic* é que as mensagens *Cyclic* são enviadas em um intervalo de tempo previamente definido, independente de terem variado ou não.

Tempo	Identificador	Grupo	MAC ID	Tamanho	Dados	Significado
04021 µs	41D	2	Destino 3	6	FF FF FF FF FF FF -- --	Master's I/O Poll Command/Change of State/Cyclic Message
00744 µs	3C3	1	Origem 3	0	-- -- -- -- -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message
09255 µs	343	1	Origem 3	6	FF FF FF FF FF FF -- --	Slave's I/O Change of State or Cyclic Message
00703 µs	41A	2	Destino 3	0	-- -- -- -- -- -- -- --	Master's Change of State/Cyclic Acknowledge Message
06165 µs	41D	2	Destino 3	6	FF FF FF FF FF FF -- --	Master's I/O Poll Command/Change of State/Cyclic Message
00739 µs	3C3	1	Origem 3	0	-- -- -- -- -- -- -- --	Slave's I/O Poll Response or Cyclic/Change of State Acknowledge Message

Figura 38 Detalhe da tela do software com mensagens Cyclic.

7.7 Avaliação de Aspectos Temporais

Além de identificar os campos das mensagens, a ferramenta desenvolvida pode ser utilizada para avaliação de aspectos temporais, sempre importantes em redes de barramentos industriais.

Com o uso da ferramenta, foi possível capturar dados referentes ao tempo de envio das mensagens. Na tela do software, pode-se verificar o tempo total entre mensagens. Para fazer uma análise da ocupação da rede, por exemplo, deve-se descontar então o tempo gasto para envio do pacote de mensagem. Esta informação auxilia na identificação real do intervalo de tempo em que a rede permaneceu ociosa.

Os dados da tabela 5 referem-se a dados temporais adquiridos durante ensaios em laboratório (ensaios descritos anteriormente). Nos dados apresentados analisam-se quatro configurações.

a) Mensagens Poll com 10 bytes de dados
b) Mensagens Poll com 8 bytes de dados
c) Mensagens COS com 8 bytes de dados
d) Mensagens Cyclic com 6 bytes de dados

Tabela 5 Tempos de ocupação e ociosidade da rede para cada tipo de mensagem.

Tipo mensagem	Quant. bytes	Tempo Ocupado	Tempo Ocioso
Poll	10	24,98%	75,02%
Poll	8	14,99%	85,01%
COS	8	1,99%	98,01%
Cyclic	6	12,44%	87,56%

A partir na análise dos tempos calculados, pode-se ver a diferença da ocupação do barramento para diferentes configurações, destacando-se a diferença entre a mensagem COS e as demais. Sabe-se que este tipo de mensagem só é enviado quando o valor do seu campo de dados é alterado ou para avisar que o dispositivo continua ativo no barramento. Logo é o tipo de mensagem mais indicado para o envio de dados que variam lentamente. Pode-se observar que, para os ensaios realizados no laboratório, a configuração COS deixa o barramento livre em torno de 98% do tempo.

Outro dado que chama a atenção para é a diferença entre as mensagens *poll* com 10 bytes e com 8 bytes. O fato de ter-se utilizado mensagens fragmentadas nos primeiros ensaios, fez a ocupação de barramento passa de perto de 15% para aproximadamente 25%, demonstrando a importância de uma rede devidamente configurada. Estes resultados podem parecer, a princípio, pouco relevantes, uma vez que mesmo no pior caso (mensagens Poll com dez bytes) a rede ensaiada ainda apresenta 75% de tempo ocioso, porém deve-se lembrar que estes ensaios foram feitos em redes com apenas duas estações ativas. Com o aumento do número de nós e de mensagens trafegadas a representação desta diferença na rede provavelmente será bem mais relevante.

Apesar de ser uma análise marginal dos aspectos temporais da rede (diversos parâmetros poderiam ser analisados, tais como tempo de ativação, tempo de resposta, tempo de aplicação fim a fim, entre outros) o que se destaca é a aplicabilidade da ferramenta para tais testes.

8 CONCLUSÕES E TRABALHOS FUTUROS

O trabalho em questão apresenta o desenvolvimento de uma ferramenta de monitoração e identificação de dados em redes industriais padrão DeviceNet. Para a implementação do mesmo, foi necessária a realização de um estudo aprofundado sobre redes de comunicação industriais, barramento CAN e mais especificamente do padrão Devicenet. Apesar de se tratarem de barramentos de campo abertos, a obtenção de informações específicas dos protocolos de baixo nível (necessário para identificação adequada das mensagens capturadas pela ferramenta) foi uma tarefa difícil, pois muitos detalhes de implementação são conhecidos por poucos profissionais especialistas da área. De fato, as informações necessárias para a realização deste trabalho só foram possíveis após a obtenção de uma versão oficial da norma DeviceNet, fornecida por representante da organização ODVA.

A validação da ferramenta desenvolvida foi feita a partir da realização de vários ensaios efetuados em uma rede de comunicação DeviceNet implementada no Laboratório de Automação Industrial da Univates, cujas configurações foram alteradas diversas vezes, para testes das funcionalidades propostas.

Como a ferramenta implementada permite captura e exibição dos conteúdos separando as partes referentes a cada campo do pacote de mensagens em redes de comunicação reais, conclui-se que a ferramenta proposta tem grande valia como mecanismo de identificação e aprofundamento do conhecimento do barramento de comunicação estudado, não somente observando o tráfego de informações (identificador da mensagem, grupo da mensagem, identificador MAC do dispositivo, tamanho do campo de dados), como também acompanhando o mecanismo de comunicação.

A utilização de uma ferramenta como essa mostra assim grande utilidade para profissionais da área, permitindo a observação *in loco* de aspectos relevantes da implementação de uma rede (que de forma alguma são evidenciados por sistemas supervisórios de alto nível). As análises temporais feitas também trazem subsídios importantes para identificação de gargalos em aplicações críticas no tempo.

Dentre os trabalhos futuros, pode-se destacar o aperfeiçoamento da parte gráfica de interação com o usuário, integrando conceitos de usabilidade e praticidade ao ambiente proposto. Novas opções de visualização, seleção de mensagens, armazenamento de dados,

assim como gráficos com informações estatísticas do barramento poderiam ser adicionados à ferramenta.

O dispositivo de aquisição implementado em hardware também deveria ser aperfeiçoado para se tornar uma solução mais flexível. Especificamente destaca-se a necessidade de se atender às três velocidades de comunicação disponíveis para este barramento (125 kbit/s, 250 kbit/s e 500 kbit/s). Além de exigir um canal de comunicação mais veloz, deveria-se também contar com uma placa com maior capacidade de armazenamento de dados, permitindo expandir os períodos de aquisição.

Outra possível atividade futura envolveria a realização de testes de validação da ferramenta de monitoração em ambientes industriais reais. Redes que utilizam um elevado número de dispositivos no barramento também apresentam grande tráfego de informações, sendo assim uma aplicação onde a utilidade de uma ferramenta como esta pode ser ainda mais relevante.

REFERÊNCIAS

BEGA, E. A.; DELMÉE, G. J.; COHN, P. E.; BULGARELLI, R.; KOCH, R.; FINKEL, V. S. **Instrumentação Industrial**. 2. ed. Rio de Janeiro. Editora Interciência: Instituto Brasileiro de Petróleo e Gás, 2006.

CiA (CAN in Automation). **CAN History**. Disponível em <http://www. 48 cia.org/index.php?id=161>. Acesso em: 15 nov. 2007a.

CiA (CAN in Automation). **CAN Physical Layer**. Disponível em <http://www.can-cia.org/index.php?id=517>. Acesso em: 15 nov. 2007b.

CiA (CAN in Automation). **CAN Specification 2.0, Part B**. Disponível em <http://www.can-cia.org/fileadmin/cia/specifications/CAN20B.pdf>. Acesso em: 15 nov. 2007c.

ERIKSSON, J.; COESTER, M.; HENNIG, C.H. **Redes Industriais – Panorama Histórico e Novas Tendências**. Controle e Instrumentação. São Paulo, n. 119, ago. 2006.

HÜSEMANN, R. **Sistema de Validação Temporal para Redes de Barramentos de Campo**. 2003. 144 p. Dissertação (Mestrado em engenharia) – Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003.

HUSEMANN, R.; PEREIRA C. E. **A multi-protocol real-time monitoring and validation system for distributed fieldbus-based automation applications.** Control Engineering Practice. United Kingdom. Vol. 15, Issue 8. Aug. 2007. Pag 955-968.

IMS RESEARSH. **Industry News**. Disponível em <http://www.anybus.com/ enews/enews_HMI07.html>. Acesso em: 13 out. 2007.

IXXAT AUTOMATION GMBH. **canAnalyser - DeviceNet Module**. Disponível em <http://www.ixxat.com/index.php?seite=devicenet_module_en> Acesso em 22 dez. 2007.

LIAN, F.; MOYNE, J. R.; TILBURY, D. M. **Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet.** IEEE Control Systems Magazine. February 2001, p. 66-83, 2001.

KIRSCHBAUM, A.; RENNER, F.M.; WILMES, A.; GLESNER, M. **Rapid-Prototyping of a CAN-Bus Controller: A Case Study**. Seventh IEEE International Workshop on Rapid System Prototyping. Proceedings, p. 146-151, 1996.

MACKAY, S.; WRIGHT, E.; REYNDERS, D.; PARK, J. **Practical Industrial Data Networks: Design, Installation and Troubleshooting**. 1. ed. London. Newnes/Elsevier. 2004.

NATALE, F. **Automação Industrial**. 8. ed. São Paulo. Editora Érica, 2000.

ODVA (Open DeviceNet Vendor Association Inc.). **DeviceNet Products – Network Configuration Tools and Diagnostics**. Disponível em <http://www.odva.org/default.aspx?tabid=84> Acesso em 22 dez. 2007a.

ODVA (Open DeviceNet Vendor Association Inc.). **DeviceNet Technical Overview**. Disponível em <http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00026R1.pdf> Acesso em 10 out. 2007b.

ODVA (Open DeviceNet Vendor Association Inc.). **Planning and Installation Manual**. Disponível em <http://www.odva.org/portals/0/library/Publications_Numbered/PUB00027R1_Cable_Guide_Print_Copy.pdf>. Acesso em 15 out. 2007c.

ODVA (Open DeviceNet Vendor Association Inc.). **The CIP Networks Library Volume 3: DeviceNet Adaptation of CIP.** 1.5 ed. Michigan. ODVA. 1CD-ROM. 2007d.

PARK, J.; MACKAY, S.; WRIGHT, E. **Practical Data Communications for Instrumentation and Control**. 1. ed. London. Newnes/Elsevier. 2003.

POPOVIC, D.; VLACIC, L. **Mechatronics in Engineering Design and Product Development**. 1. ed. New York. Marcel Dekker Inc., 1999.

ROSÁRIO, J. M. **Princípios de Mecatrônica**. 1. ed. São Paulo. Prentice Hall, 2005.

SOARES, L. F. G.; LEMOS, G.; COLCHER, S. **Redes de Computadores**. 6. ed. Rio de Janeiro. Editora Campus, 1995.

TANENBAUM, A. S. **Redes de Computadores**. 4. ed. Rio de Janeiro, Brasil: Ed. Campus, 2003.

VECTOR INFORMATIK. **CANalyzer - CANalyzer.DeviceNet**. Disponível em <http://www.vectorworldwide.com/portal/medien/cmc/datasheets/CANalyzer.DeviceNet_DataSheet_EN.pdf> Acesso em 23 dez. 2007.

WOODHEAD INDUSTRIES. **DeviceNet Network Analyzer**. Disponível em <http://www.mysst.com/pub/products/devicenet/DN3/7170025_DeviceNet%20Network%20Analyzer%20User%20Guide.pdf> Acesso em 23 dez. 2007.