UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE REDE INDUSTRIAL UTILIZANDO O PADRÃO SERIAL RS485 E PROTOCOLO MODBUS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

JUDSON MICHEL CUNHA

BLUMENAU, DEZEMBRO/2000

2000/2-31

PROTÓTIPO DE REDE INDUSTRIAL UTILIZANDO O PADRÃO SERIAL RS485 E PROTOCOLO MODBUS

JUDSON MICHEL CUNHA

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____
Prof. Sérgio Stringari – Orientador da FURB

_____
Prof. José Roque Voltolini da Silva – Coordenador do TCC

BANCA EXAMINADORA

_____
Prof. Antonio Carlos Tavares

_____
Prof. Miguel Alexandre Wisintainer

_____
Prof. Sérgio Stringari

“Não existe o caminho para a felicidade, pois ela é o próprio caminho”

“À minha mãe Ingrit, minha irmã Lenaide e minha vó Lúcia, pelo amor e dedicação a minha formação pessoal e incentivo à busca do conhecimento”

AGRADECIMENTOS

Muitas pessoas contribuíram para a realização desse trabalho, às quais quero agradecer de maneira especial.

À minha família que estão sempre ao meu lado, pelo seu incentivo, pela sua dedicação à minha formação, e pela vida familiar e afetiva tanto nos momentos difíceis quanto nos bons momentos da minha vida.

Ao professor Sérgio Stringari, meu orientador e amigo, que me apoiou e sempre esteve disposto a me ajudar e esclarecer as dúvidas.

Aos meus amigos, pessoas importantes que de alguma maneira ajudaram-me a superar obstáculos e seguir em frente.

A Paulo Lange, diretor da empresa DWA, pela sua disponibilidade e apoio no esclarecimento das dúvidas e desenvolvimento do trabalho. Pela sua compreensão aos momentos em que tive que me ausentar da empresa.

A alguém muito especial em minha vida. Pelo seu amor, seu afeto e seu companheirismo. E por tudo aquilo que as palavras não conseguem expressar...

A todos que de alguma forma contribuíram para minha formação acadêmica e pela elaboração desse Trabalho de Conclusão de Curso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

PC – Personal Computer

RS – Recommended Standard

JIT – Just in Time

TQM – Gerenciamento Total da Qualidade

CLP – Controlador Lógico Programável

DLL – Dinamic Link Library

API – Application Program Interface

TX – Transmissão

RX – Recepção

CNC – Controle Numérico Computadorizado

ISO - International Standard Organization

Mbps – Megabits por segundo

Ms – milisegundos

INMETRO – Instituto de Metrologia

CI – Circuito Integrado

Bps – bits por segundo

HCB – Host Controller Board

LCR – Longitudinal Redundancy Check

CRC – Cycling Redundancy Check

CR – Carry Return

PPM – Pacotes por minuto

GLOSSÁRIO

Layouts – Forma de organizar os objetos em um espaço físico

Reset – Processo informatizado para exclusão de dados

Array – Lista encadeada ou não encadeada de variáveis computacionais

RESUMO

Este trabalho tem por objetivo principal, especificar e implementar um protótipo de sistema para comunicação serial entre controladores de peso CHECK 5P, da empresa DWA, utilizando os padrão RS232C e RS485 como meio de transmissão serial e o protocolo Modbus, para troca de mensagens entre os controladores de peso e o microcomputador. O protótipo, irá disponibilizar os dados sobre o processo de empacotamento, armazenados em memória RAM dos controladores de peso, para serem processados e analisados por softwares de programação e controle de produção, que sigam as especificações do protótipo, como forma de aquisição de dados. Como objetivos secundários, será apresentado um estudo sobre as tecnologias de redes industriais, o protocolo Modbus e os padrões seriais RS232C e RS485.

ABSTRACT

The main objective this work is to specify and to implement a system prototype to serial communication between checkweigh Check 5PA, by DWA Company, using the RS232C and RS485 standard like way to do serial transmition and the Modbus protocol, to change the messages between checkweighers and computer. The prototype will be to arrange the datas about packaging process, stored in RAM memory from checkweighers, to be processed and analysed for the programation softwares and prodution control, it follow the prototype specifications, like way datas acquisition. It will be show like secondary objectives a study about industries nets tecnology, the Modbus protocol and the RS232C and RS485 serial standard.

1 INTRODUÇÃO

Hoje em dia, vê-se na grande maioria das empresas, que muitos processos, principalmente os burocráticos, estão informatizados para se obter uma maior agilidade, tanto na entrada das informações, quanto na recuperação da mesma ([MAR20000]).

Um problema enfrentado por muitas empresas, no que se refere a automatização da obtenção de dados, está nas informações geradas no chão de fábrica, ou seja, nos equipamentos de produção. Especificamente pode-se citar as máquinas de empacotamento e controle de peso ([URB1993]).

Algumas empresas de automação, percebendo o nicho de mercado, favorecidos também pela legislação, investiram em controladores de peso, o que não é simplesmente uma balança, pois as balanças somente pesam, e os controladores, fazem a dosagem certa para que os produtos dentro do pacote, tenham pesos condizentes com a lei. Como exemplo, cita-se a empresa DWA, referindo-se em controladores de peso, e seu controlador de peso CHECK 5P.

Como tratava-se de equipamentos de alta tecnologia e igualmente caros, começou-se a pensar sobre algumas utilidades que o equipamento pudesse fornecer, além, é claro, do controle do peso ([URB1993]). Pode-se citar o armazenamento de quantidades produzidas, programação remota do equipamento, interligação desses equipamentos com um Computador Pessoal (PC), além de outras.

Dentre as várias formas de interligar dois ou mais equipamentos computacionais, utiliza-se mais comumente a comunicação serial ou paralela dos atuais PC´s ([TAN1994]). Segundo Tanenbaum ([TAN1994]), um PC utiliza a tecnologia RS232C, (*Recommended Standard*) definida da estrutura IBM-PC, como forma de acesso ao meio físico. O RS485 é um dos padrões de comunicação serial e bidirecional utilizados em aplicações industriais. Os padrões seriais RS485 e RS232C definem as características elétricas para cada interface ([KLI1999]).

Outro aspecto, na comunicação serial entre equipamentos, é o protocolo que irá efetuar a conexão e troca de mensagens entre eles. Como modelo de estudo, seguir-se-á, a proposta do *Modbus* no modo difusão, que segundo Morais ([MOR2000]), “os equipamentos

são os escravos e a sua atividade na rede consiste em responder às questões emitidas pelo PC, ou seja, o mestre. O mestre pode questionar um escravo em particular e esperar pela sua resposta (modo questão/resposta), ou, pode enviar uma ordem comum a todos os escravos (modo difusão)".

1.1 OBJETIVOS

Este trabalho tem por objetivo principal, especificar e implementar um protótipo de sistema para comunicação serial entre controladores de peso CHECK 5P, da empresa DWA, utilizando os padrões RS232C e RS485 como meio de transmissão serial e o protocolo *Modbus*, para troca de mensagens entre os controladores de peso e o microcomputador.

Para a realização desse trabalho, deve-se considerar outros objetivos tidos como secundários:

a) estudo da máquina de empacotar e controlador de peso, para compreender o processo e a contextualização do problema;

b) estudo dos padrões seriais RS223C e RS485;

c) estudo do protocolo *Modbus*.

1.2 ORGANIZAÇÃO DO TEXTO

Para um melhor entendimento, este trabalho está organizado em 7 capítulos.

O capítulo 1 descreve os objetivos desse trabalho, bem como trata-se de uma introdução às redes industrias.

O capítulo 2 descreve as redes industriais, as tecnologias disponíveis no mercado e a forma de funcionamento delas.

O capítulo 3 descreve o protocolo implementado no desenvolvimento desse trabalho, o *Modbus*.

No capítulo 4 ter-se-á uma explanação mais específica sobre os padrões RS232C e RS485, a compatibilidade entre eles, suas especificações e sua implementação a nível de mercado.

No capítulo 5 será descrito o ambiente de trabalho. Será apresentada a máquina de empacotar e sua forma de funcionamento e também o controlador de peso, com suas características e forma de funcionamento.

No capítulo 6 será abordado o protótipo a nível de especificação e implementação. É mostrado o funcionamento interno do protótipo, as ferramentas utilizadas na especificação e o fluxograma dos processos. Na implementação serão apresentadas também todas as telas do protótipo e sua forma de utilização, a nível de usuário, como também algumas funções relevantes às telas descritas.

Por fim, no capítulo 7 conclui-se o trabalho com a exposição das idéias relativas ao mesmo, resultados alcançados e resultados esperados, e a avaliação do protótipo.

2 REDES INDUSTRIAIS

Segundo Delgado ([DEL2000]), desde os tempos mais remotos, o homem quer libertar-se do trabalho muscular, animal e das tarefas pesadas. Para isso vem-se desenvolvendo estratégias e mecanismos que permitam essa finalidade. Juntamente com essas estratégias, vem conseguindo maior velocidade na execução das tarefas, menores tempos de paradas, menor número de acidentes e a obtenção de produtos em quantidade maior e maior qualidade.

O objetivo foi desde sempre fazer a dosagem correta dos três componentes fundamentais no processo produtivo, que são eles: a matéria, a informação e a energia.

De início, a automatização era caracterizada por pequenas ilhas com operações automatizadas, onde o fator humano era fundamental como elemento integrador e sincronizador de todas as operações. Este estágio caracterizava-se, por um elevado número de operários, a existência de grandes estoques e *layouts* não otimizados.

Caminhou-se depois para soluções de automatização centralizada. Nestas, toda a informação é centralizada num único local, onde são tomadas todas as decisões e de onde partem todas as ordens. Com este nível, os *layouts* foram melhorados, o número de operários bastante reduzido, mas continua a existir um nível considerável de estoque.

Após a década de 60, com o desenvolvimento e a utilização crescente de unidades de processamento de informação, as funções de condução dos processos foram sendo cada vez mais distribuídas pelo terreno e junto dos locais onde são necessárias, surgindo assim o que é atualmente designado por *Arquiteturas Distribuídas* ou por *Sistemas de Controle Hierárquico Distribuído*. Este nível de automatização caracteriza-se por uma gestão global e integrada da informação, pela redução de estoques a níveis mínimos, pela inserção de CNCs (controladores numéricos computadorizados), de manipulação (Robôs), pela redução drástica do número de operários, sendo em alguns setores praticamente nulo na área diretamente relacionada com a produção, pela utilização dos modernos conceitos de JIT (*Just-in-Time*) e TQM (*Total Quality Management*) e ainda por uma utilização muito mais intensiva dos equipamentos.

Para que esses conceitos fossem aplicados, e não se perdesse o controle principalmente da informação e operação, segundo Lima ([LIM2000]), “um dos fatores importantes na empresa é a rede industrial de comunicação de dados, pois ela é a responsável pela troca de informações e a sincronização entre os processos envolvidos”.

Algumas características das redes industriais são apresentadas abaixo:

a) o ambiente é hostil para operação de equipamentos. Há muitas perturbações eletromagnéticas, elevadas temperaturas, sujeira, etc.;
b) a troca de informações ocorre entre equipamentos e não entre um operador e o equipamento;
c) os tempos de resposta e a segurança dos dados são críticos em diversas situações;
d) uma grande quantidade de equipamento pode estar conectada na rede;
e) há tempo de acesso ao meio físico;
f) na transmissão de códigos de comando, leitura de medidores e comando de atuadores, um erro em um bit pode ter conseqüências desastrosas.

Vencidas essas barreiras, grandes avanços ocorreram a nível de chão de fábrica, onde os equipamentos CNCs, CLPs (Controlador Lógico Programável), entre outros começaram a comunicar-se com o mundo exterior. Porém, cada fabricante definia um protocolo proprietário e a integração destes sistemas tornou-se impraticável.

Para solucionar estes problemas, muitas empresas começaram a desenvolver projetos de arquiteturas para ambientes industriais, que integrassem sistemas heterogêneos de diferentes fabricantes, suportando tanto as operações de chão de fábrica quanto as funções de apoio à produção ([LIM2000]). Desta forma, equipamentos produzidos por fabricantes diferentes poderiam ser facilmente incorporados à instalação, simplesmente conectando-os ao sistema de comunicação.

Atualmente no Brasil, com a globalização da economia e o aumento da competitividade, as empresas estão sendo obrigadas a adquirir um alto grau de aperfeiçoamento dos processos e o gerenciamento das informações, possibilitando o aumento da produtividade e a redução dos custos. Os níveis de qualidade dos processos devem ser

garantidos através de normas técnicas, operacionais e auditorias periódicas que permitem o reconhecimento através dos certificados ISO (International Standard Organization) .

Através dos sistemas de automação industrial e redes industriais é possível controlar cada etapa dos processos, fazendo com que a própria aplicação defina quais são os procedimentos adotados para cada tarefa e monitore grande parte dos trabalhos dos operadores.

Um dos problemas na obtenção e gerenciamento das informações é a dificuldade de se alimentar os sistemas com dados, obtidos normalmente através de um processo de digitação. Essa maneira de entrar com as informações nos computadores é um processo muito lento e apresenta o grande inconveniente de estar sujeito a erros. Segundo Ribeiro ([RIB2000]), “pior que não se coletar a informação é coletar essa informação com erros, o que pode distorcer os relatórios gerenciais e dificultar a análise dos resultados da empresa e na tomada de decisões”.

Esta crescente automação nas indústrias brasileiras vem sendo impulsionada principalmente por três fatores:

a) as empresas sentem cada vez mais a necessidade de redução de custos;
b) os preços dos equipamentos e sistemas vem caindo significativamente;
c) a capacidade destes mesmos sistemas vem crescendo em progressão geométrica.

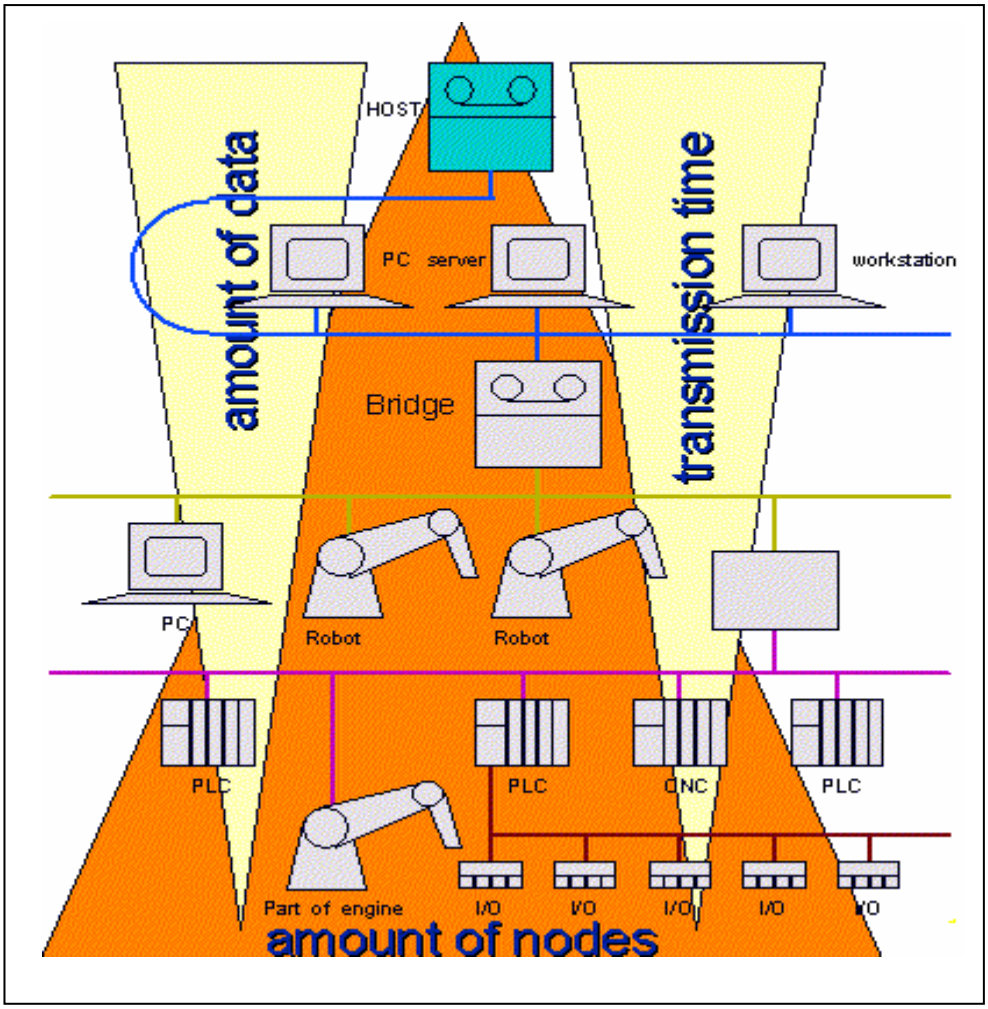
2.1 NÍVEIS DAS REDES INDUSTRIAIS

As redes de comunicação industrial, basicamente se caracterizam pela relação informações versus tempo. As diversas propostas para se interligarem equipamentos industriais com computadores que tratam as informações, buscam sempre estruturas que garantam a segurança na transmissão dos dados, e a velocidade de comunicação ([SIL2000]).

Um ponto importante a salientar, é que uma única rede, atende normalmente aos diversos níveis propostos. Por exemplo, pode-se citar um controlador programável, que comanda dispositivos como sensores, atuadores, etc, no nível de dispositivos de campo, e ao mesmo tempo, fornece informações para um sistema supervisório no nível de informação.

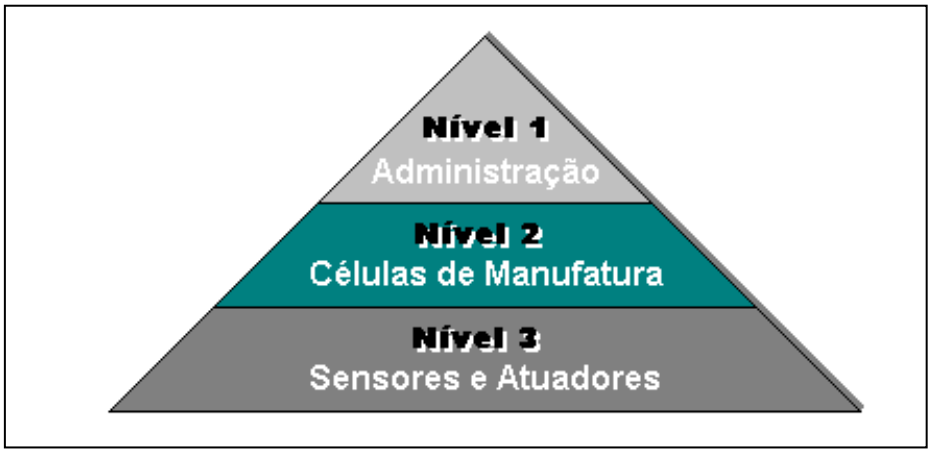
Outro aspecto é a velocidade das redes industriais. Costuma-se medir a velocidade pela taxa de transmissão de bits (meio físico), depois pelo protocolo utilizado e o mecanismo de troca de dados. Estes, podem ser tanto softwares de controle implementados em PC´s ou em CLP´s. Porém, o principal responsável pelo desempenho, é inverso ao apresentado acima, Não adianta comunicar a altas velocidades, com informações mal dispostas ou redundantes. Na figura 2.1, pode-se observar melhor a distribuição dos dispositivos em uma rede industrial, bem como o montante de dados envolvidos e tempos em cada nível. Com maior quantidade de dados e maior tempo temos os servidores, estações de trabalho (PC) e mestres da rede, abaixo já reduzindo a quantidade de dados e tempo, tem-se os controladores de processos (Robôs e as vezes CLP). Como terceiro nível e com quantidade de dados reduzida e tempos menores, tem-se os atuadores, que são dispositivos responsáveis por alguma tarefa específica.

Figura 2.1 – Distribuição dos dispositivos em um rede industrial ([MON2000])



Na figura 2.2 são apresentados os níveis das redes industriais.

Figura 2.2 – Níveis de uma rede Industrial ([SIL2000])



2.1.1 O PRIMEIRO NÍVEL (ADMINISTRAÇÃO)

No topo da pirâmide (primeiro nível) está situada a empresa através de uma visão macro, tendo todas as ações da rede de comunicação dirigidas para o controle gerencial da produção. Neste nível estão envolvidos a administração, gerência, contabilidade, compras, vendas, produtividade, estratégias de ação, banco de dados, entre outros. Os controladores presentes neste nível são, basicamente, PC's na arquitetura padrão de Cliente-Servidor, distribuídos em topologias das mais diversas, como: barra, anel, árvore ou outras. Segundo Silverado ([SIL2000]), em termos de comunicação é neste nível que a quantidade de informações é gigantesca, porém, os desenvolvimentos atuais de redes de comunicação padrão *Ethernet*, *Fast-Ethernet* ou outros, conseguem atingir um tempo de resposta de até 1000ms (milisegundos) à uma velocidade de comunicação que pode variar de 10 a 100Mbps (mega bits por segundo).

2.1.2 O SEGUNDO NÍVEL (CÉLULAS)

No segundo nível estão localizados os controladores que gerenciam processos, linhas de montagens ou mesmo máquinas automáticas. A comunicação neste nível é feita entre os mestres ou gerentes das células. Podem ser citados também os CNC´s, PC's utilizados em controle de processos ou CLP's no comando de máquinas automáticas. Estes diversos

controladores se comunicam entre si através de um protocolo padrão definido no projeto da rede, e a comunicação com o nível mais superior (administração) é feito através de um equipamento (PC) colocado estrategicamente na fábrica. Esse PC faz a coleta das informações necessárias a ser transmitidas ao nível superior. Neste segundo nível as redes caracterizam-se por possuírem uma quantidade de informações (tráfego) média, o que resulta num tempo de resposta nunca maior do que 100ms a uma velocidade que pode chegar até 12Mbps.

2.1.3 O TERCEIRO NÍVEL (SENSORES E ATUADORES)

No nível mais baixo da pirâmide está localizado o barramento de campo, que é responsável pela comunicação entre os dispositivos mais simples, utilizados no chão da fábrica (sensores e atuadores) e seus respectivos controladores. A comunicação neste nível é feita da seguinte forma: um dos controladores acima citado possui um mestre de rede, este mestre porém, possui escravos aos quais são conectados os sensores e atuadores inteligentes ou convencionais presentes no sistema. A forma física de conexão ou a topologia empregada depende de cada tipo de barramento. Este nível se caracteriza, em termos de comunicação, por possuir uma quantidade de informações baixa trafegando na rede, trabalhando com velocidades e taxas de transmissão baixas que chegam até 2Mbps e tempo de resposta de aproximadamente 10ms ([MON2000]).

2.2 ESTRUTURA DE COMUNICAÇÃO

Partindo-se agora para a forma de comunicação dos equipamentos em uma rede industrial, verifica-se que a camada de enlace, responsável pelo mecanismo de entrega de pacotes, segundo modelo OSI, tem sido implementada tradicionalmente em redes industrias com a estrutura *origem/destino e produtor/consumidor*. Também podem ser chamadas de estruturas *mestre/escravo ou modo difusão* ([ROG2000]).

Na estrutura *mestre/escravo*, cada nó da rede tem uma identificação, para a qual o equipamento mestre da rede envia os dados para um nó específico. Sendo assim, envia-se a mensagem ao nó da rede e espera-se pelo retorno. A implementação, dessa estrutura, torna-se ineficiente quanto se deseja transmitir um mesmo dado para mais de um nó da rede,

pois como ela trabalha com o endereçamento deve-se mandar a mesma informação para todos os nós da rede, aumentando o tráfego da rede e constituindo um operação repetitiva.

Segundo Moraes ([MOR2000]), o mestre tem quatro atividades na rede:

a) assegurar a troca de informação entre os escravos. Como os escravos não podem se comunicar entre si, o mestre irá coordenar o tráfego de informações;

b) assegurar a troca de informações entre homem/máquina, com uma interface gráfica para a entrada e saída de informações, visualizando assim o processo que se deseja controlar;

c) assegurar o diálogo com os demais mestres da rede, ou mestres de outras redes;

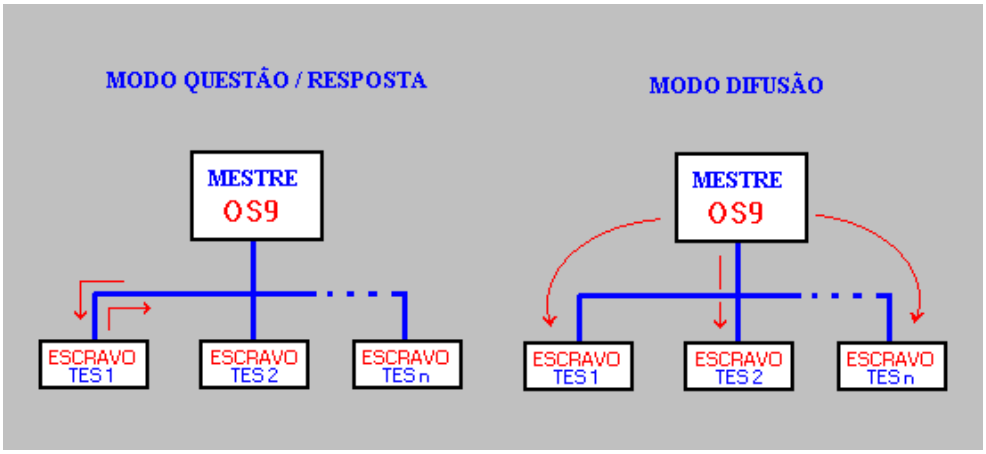
d) assegurar a passagem de parâmetros de programação para os escravos a fim de obter-se maior agilidade e menor perda no processo automatizado.

Já na estrutura *modo difusão* a implementação está baseada no conceito de que alguns dispositivos são produtores de informações e outros são consumidores dessas. Nessa implementação, quando um produtor disponibiliza sua informação, esta é colocada na rede, disponível para todos os dispositivos que sejam seus consumidores ao mesmo tempo, reduzindo o número de mensagens a serem emitidas, bem como reduzindo o próprio comprimento da mensagem, uma vez que não será necessário incluir os endereços de remetente e destinatário, sendo necessário tão somente identificar a informação a ser transmitida.

Porém, cada estrutura tem sua necessidade de implementação, conforme a aplicação que se for utilizar. De acordo com Rogana ([ROG2000]), “o modelo produtor/consumidor, está hoje sendo empregado em produtos comerciais para redes industriais, tais como *Foundation Fieldbus*, *WorldFIP*, *ControlNet* e *DeviceNet*. Outros produtos podem ainda utilizar as duas estruturas ao mesmo tempo”.

A figura 2.3 ilustra a estrutura de comunicação mestre/escravo (modo questão resposta) e a estrutura difusão.

Figura 2.3 – Modelo das Estruturas da Rede Industrial ([COT2000])



2.3 TOPOLOGIAS

A Topologia da uma rede, segundo Rombaldi ([ROM2000]), trata da distribuição geográfica de elos e nós da rede. Um elo ou ramo é uma trajetória de comunicação entre dois nós. O termo elo é usado como sinônimo de canal e circuito, e pode ser de vários tipos, tais como rádio, fibra ótica, satélite, cabo coaxial ou linha telefônica. Já um nó pode ser definido como qualquer ponto terminal de qualquer ramo da rede, ou a junção de dois ramos quaisquer. O hardware e o software de um nó depende de sua função principal.

Existem dois tipos básicos de rede: ligação ponto-a-ponto e multiponto. Combinando-se os dois tipos básicos formam-se redes mais complexas, as chamadas estruturas mistas.

2.3.1 PONTO-A-PONTO

Nesta rede, o nó central é conectado a um equipamento de comunicação de entrada e saída por uma única linha. Sempre que algum nó tiver algo a transmitir, a linha estará livre, já que não há compartilhamento com outro nó do canal. Nesta modalidade de ligação, o hardware conectado ao mestre é o mais simples possível e o software de atendimento não precisa ser muito sofisticado.

2.3.2 MULTI-PONTO

Nesta modalidade de ligação existe sempre uma estação controladora que coordena o tráfico de dados das demais estações chamadas subordinadas. Este controle é feito através de uma rotina de atendimento denominada "*POLL-SELECT*", ou multiplexando-se o canal de comunicação, que dispensa o controle de endereçamento do "*Poll-Select*" ([OLI1987]) . Estas redes podem permitir que estações subordinadas se comuniquem entre si diretamente ou apenas através da estação controladora. A diferença entre estes dois modos de envio de mensagens é a complexidade do controle.

2.3.3 ESTRUTURAS MISTAS

As Estruturas Mistas são tipos de redes que utilizam características dos dois tipos básicos de redes, a ligação ponto-a-ponto e multiponto, para obter redes mais complexas e com maiores recursos. As estruturas mistas podem ser do tipo Estrela, Barra, Hierárquica, Anel e Distribuídas.

2.3.3.1 ESTRELA

Neste tipo de rede, todos os usuários comunicam-se com um nó central, que tem o controle supervisor do sistema, chamado "*host*". Através do *host* os usuários podem se comunicar entre si e com processadores remotos ou terminais. No segundo caso, o *host* funciona como um comutador de mensagens para passar os dados entre eles. O arranjo em estrela é a melhor escolha se o padrão de comunicação da rede for de um conjunto de estações secundárias que se comunicam com o nó central. Como exemplo dessa topologia tem-se as Redes Públicas de Telefonia Comutada ([SOA1993]).

2.3.3.2 BARRA

Nesta configuração todos os nós se ligam ao mesmo meio de transmissão. A barra é geralmente compartilhada em tempo e freqüência, permitindo a transmissão de informação. Nas redes em barra comum, cada nó conectado à barra pode ouvir todas as informações transmitidas. Esta característica facilita as aplicações com mensagens do tipo difusão (para múltiplas estações).

Nas topologias em barra, as falhas não causam a parada total do sistema. Relógios de prevenção ("*watch-dog-timer*") em cada transmissor devem detectar e desconectar o nó que falha no momento da transmissão. O desempenho de um sistema em barra comum é determinado pelo meio de transmissão, número de nós conectados, controle de acesso, tipo de tráfego entre outros fatores. O tempo de resposta pode ser altamente dependente do protocolo de acesso utilizado.

2.3.3.3 HIERÁRQUICA

Segundo Rombaldi ([ROM2000]), a topologia Hierárquica ou em árvore é essencialmente uma série de barras interconectadas. Geralmente existe uma barra central onde outros ramos menores se conectam. A ligação entre barras é realizada através de derivadores e as conexões das estações realizadas da mesma maneira que no sistema de barras padrão.

Esta topologia é muito usada para supervisionar aplicações de tempo real, como algumas de automação industrial e automação bancária. Pequenos sistemas baseados em mini ou microcomputadores proporcionam o atendimento em tempo real das atividades da agência bancária. Quando uma operação exige acesso a informações que não estão disponíveis na agência, elas são buscadas no computador central. Se este não tiver acesso direto a estas informações, redirecionará a busca para outro computador da rede que as detém.

2.3.3.4 ANEL

Uma rede em anel consiste de estações conectadas através de um caminho fechado. Nesse tipo de configuração, a idéia é que se possa ter acesso a cada ponto da rede por dois lados, dando uma maior confiabilidade ([OLI1987])

Redes em anel são capazes de transmitir e receber dados em qualquer direção, mas as configurações mais usuais são unidirecionais, de forma a tornar menos sofisticados os protocolos de comunicação que asseguram a entrega da mensagem corretamente e em seqüência ao destino ([ROM2000]).

Quando uma mensagem é enviada por um nó, ela entra no anel e circula até ser retirada pelo nó destino, ou então até voltar ao nó fonte, dependendo do protocolo empregado. O último procedimento é mais desejável porque permite o envio simultâneo de um pacote

para múltiplas estações. Outra vantagem é a de permitir a determinadas estações recebem pacotes enviados por qualquer outra estação da rede, independentemente de qual seja o nó destino.

2.3.3.5 DISTRIBUÍDAS

Esta configuração consiste de vários pontos de concentração, cada um com seu conjunto próprio de terminais geograficamente concentrados. As ligações são estabelecidas apenas entre estes pontos de concentração, o que diminui consideravelmente o custos das linhas. Só estas linhas precisarão ter uma capacidade muito maior de transmissão para poder atender às requisições de comunicação exigidas pelos seus terminais.

Para se garantir que, em caso de falha de linhas entre pontos centralizadores, as transmissões não serão interrompidas, é comum a conexão destes centros a mais de um outro centro. Outra forma de redundância de linhas é a conexão de cada ponto central a todos os demais pontos de concentração. Nesta rede, denominada completamente conectada, a probabilidade de estrangulamento nos horários de pico de tráfego é muito baixa e sua confiabilidade é muito maior. O problema é o altíssimo custo das linhas.

2.4 MEIOS DE TRANSMISSÃO

Segundo Franco ([FRA2000]), *fieldbuses* são tecnologias de comunicações e protocolos usados em automação e controle de processos industriais, formando uma rede industrial. Pode-se distinguir entre *fieldbus* proprietário e aberto. Os *fieldbus* proprietários são aqueles que não temos acesso a sua tecnologia, o que é inverso no *fieldbus* aberto.

Os meios de comunicação entre os equipamentos na rede industrial, são geralmente os meios seriais, ou seja, os padrões de RS (*Recommended Standard*). Esses padrões surgiram em meados de 1962, pela necessidade de se interligar dois ou mais pontos em uma rede de computadores. Sua especificação foi criada *pela Eletronics Industry Association* (EIA), e definiu-se como sendo RS as siglas que a designariam. Outros meios são as fibras óticas, que pouco usadas oferecem uma taxa de transmissão altíssima e as maiores velocidades do mercado. Outros meios físicos de transmissão de dados usados em redes industriais, são os pares trançados e cabos coaxiais. A meio coaxial e tecnologia Ethernet, está no momento sendo implementado de forma ampla, graças a sua integração com o protocolo TCP/IP. Abaixo, serão descritos os padrões seriais de forma mais detalhada.

2.4.1 PADRÕES SERIAL

O padrão serial RS (*Recommended Standard*), define-se em dois modos de transmissão: O *Single-Ended Data Transmission* e o *Differential Data Transmission*.

O primeiro modelo o *Single-Ended* é caracterizado pela comunicação de uma ponta a outra. Ou seja, em uma rede com um *mestre* e *n* escravos, deve sair do dispositivo mestre um cabo para cada escravo. Outro detalhe do modo *Single-Ended*, é que os dados são representados em níveis de voltagem em relação ao um terra comum. Esse sistema torna a rede serial lenta, aproximadamente 20 Kbits/s (kilobits por segundo) de transmissão e a curtas distâncias de aproximadamente 15 metros sem repetidores. O modo *Single-Ended* é encontrado nos padrões RS232C e RS423 ([ARC2000]).

O Segundo modelo, o *Differential Data Transmission*, oferece uma alta taxa de transmissão, aproximadamente 100 Kbits/s e longas distâncias, chegando até 1200 metros. Usa-se uma linha diferenciada, o que implica que o dado é representado pela corrente e não

pela voltagem como no modelo *Single-Ended*. O *Differential* possibilita, pela sua estrutura, uma conexão multi-ponto, onde pode-se ter uma conexão de um mestre e vários escravos compartilhando mesmo meio. Essas implementações podem ser encontradas nos padrões RS422 e RS485 ([ARC2000]).

2.4.1.1 RS232C

O padrão RS232C, possui canais independentes de transmissão, ou seja, uma linha para transmissão e outra para recepção de dados. Os dados são representados por sinais através de níveis de voltagem. Segue o modelo de comunicação *Single-Ended*, e possui uma taxa de transmissão relativamente baixa, chegando até 20Kbps a uma distância de 15 metros. É comumente usada para conexão entre dois equipamentos.

2.4.1.2 RS422

O padrão RS422, permite a conexão de equipamentos a longas distâncias e altas taxas de transmissão em relação ao RS232C. Isso por que utiliza o modelo de transmissão *Differential*, possibilitando a transmissão a 10 Mbps e a uma distância de 1200 metros. Além de possibilitar uma conexão multi-ponto, com um mestre e 10 receptores.

2.4.1.3 RS423

A RS423 assim como a RS422 permite a conexão a longas distâncias, aproximadamente 1200 metros e altas taxas de transmissão, chegando até 100Kbps. Sua velocidade é inferior a RS422, por utilizar o modelo *Single-Ended*. Permite conexão multi-ponto, com um mestre e 10 receptores na rede.

2.4.1.4 RS485

A RS485 diferencia-se dos demais modelos, por utilizar um ou dois pares de fios para transmissão de dados. Tem isolamento óptico, e trabalha a taxas de transmissão de 10 Mbps e aproximadamente 1200 metros de distância, sem amplificação do sinal. Permite a conexão multi-ponto, suportando até 32 equipamentos na rede, ou seja, um mestre e 31 escravos. Utiliza o modelo *Differential* para transmissão.

2.5 TÉCNOLOGIAS DE REDES INDUSTRIAIS

A quantidade de padrões existentes para uma rede de campo é muito grande, diferenciando-se cada um deles pela aplicação, quantidade e velocidade de transmissão dos dados, disposição e meio físico da rede e o protocolo de comunicação utilizado. Abaixo estão relacionados alguns padrões utilizados em redes industriais ([SIL2000]).

2.5.1 AS-I (ACTUATOR SENSOR INTERFACE)

AS-I é um sistema de rede de sensores e atuadores de baixo nível. Normalmente os sinais dos sensores e atuadores dos processos industriais são transmitidos através de um grande número de cabos. O sistema ASI permite a simplificação deste sistema de fiação e ligação, substituindo o então sistema rígido de cabos por apenas um par de fios, que podem ser usados por todos os sensores e atuadores. Este par de fios é responsável pela alimentação dos sensores/atuadores e pela transmissão dos dados binários de entrada e saída. A rede foi concebida para complementar os demais sistemas e tornar-se mais simples e rápida a conexão dos sensores e atuadores com os seus respectivos controladores.

O padrão começou a ser desenvolvido por uma associação de fabricantes europeus em 1990 que se propôs a conceber uma rede de comunicação, de baixo custo, e que atendesse o nível mais baixo da automação no campo. O término dos trabalhos ocorreu em 1993. Posteriormente este grupo foi desfeito e a tecnologia passou a ser administrada por uma Associação Internacional (*AS - International*), fazem parte desta associação sessenta e cinco membros, situados em nove países. Os nove países que possuem fabricantes de produtos com tecnologia ASI são os seguintes: Bélgica, Franca, Alemanha, Inglaterra, Itália, Japão, Holanda, Suíça e Estados Unidos. Esta associação é responsável por determinar os critérios de padronização dos produtos, especificações técnicas, testes dos produtos, juntamente com a divulgação e o marketing da tecnologia. Atualmente existem cerca de duzentos produtos com tecnologia ASI no mercado.

O sistema baseia-se numa comunicação mestre-escravo, cujo mestre é responsável pelo direcionamento das "perguntas" e tratamento das "respostas" dos escravos. O mestre pode gerenciar até trinta e um escravos. A comunicação entre o mestre e os escravos é feita serialmente através de um par de fios não trançados e nem blindados. Inicialmente o mestre "fala" com o primeiro escravo, atualiza as saídas do mesmo (se existir) e pergunta o estado binário das entradas. Imediatamente o escravo responde e, após uma pequena espera, o mestre

"fala" com o próximo escravo. Após o escravo trinta e um, o ciclo se completa e o mestre começa a conversar novamente com o escravo numero um. O ciclo de varredura completo tem duração de até 5ms (contendo 31 escravos na rede). Um escravo caracteriza-se por possuir um *chip* (*ASIC - Application Specific Integrated Circuit*) especialmente desenvolvido e que possui quatro bits que podem ser configurados como entrada ou saída. Este *chip* também é responsável por determinar o endereço de cada escravo. O procedimento de endereçamento dos escravos é feito através de unidade de endereçamento. Os sensores ou atuadores "burros", ou seja, que não são considerados escravos (não possuem o *chip*) podem ser conectados à rede através de módulos de entrada e saída.

A rede ASI permite o uso de múltiplos tipos de topologias de rede, permitindo que a qualquer momento se possa iniciar uma nova derivação, possibilitando a inclusão de novos sensores e atuadores. Cada usuário pode escolher sua topologia conforme sua necessidade e disposição física dos elementos no campo. O cabo da rede não necessita de resistor nno último ponto da rede, para terminação da mesma, sua única limitação está relacionada com o comprimento do fio, que deve possuir cem metros de comprimento. Caso necessário, o cabo pode ter um acréscimo de duzentos metros com a utilização de repetidores (*boosters*), ficando assim, com um comprimento total de trezentos metros

2.5.2 PROFIBUS (PROCESS FIELD BUS)

O padrão *Profibus* subdivide-se em três famílias: *Profibus-DP*, *Profibus-PA e v Profibus-FMS*. A terceira família está situada no segundo nível da pirâmide (*cell level*), já as outras duas estão voltadas para o *fieldbus*.

2.5.2.1 PROFIBUS-DP

Esta família foi desenvolvida em 1994, para fazer a comunicação entre os sistemas de controle (controladores) e os elementos de campo através da configuração mestre-escravo. O sistema pode ser configurado como *mono-master* (apenas um mestre) *ou multi-master* (com vários mestres), sendo que neste último, as entradas podem ser lidas por todos os mestres e cada mestre acionando apenas suas respectivas saídas. A topologia utilizada é em linha,

utilizando o par trançado ou fibra óptica como meio físico. A transmissão dos dados é feita através de RS485 e a taxa de transmissão está relacionada com a distância do cabo (9,6 Kbits 1200m, 500 Kbits 400m, 12000 Kbits 100m por exemplo). O sistema comporta 32 estações sem a utilização de repetidores e até 127 estações com a utilização de repetidores. Quando do término do meio físico da rede, a mesma necessita da colocação de um terminador de rede (resistor de terminação), responsável por garantir a imunidade à ruídos e determinar o final da rede.

2.5.2.2 PROFIBUS-PA

Esta família foi desenvolvida em 1995, de acordo com a norma IEC 1158-2, a qual é utilizada na automação e controle de processos contínuos, principalmente no setor químico e petroquímico. O Profibus-PA permite que os sensores e atuadores sejam conectados ao cabeamento mantendo a segurança intrínseca dos elementos requerida pelo processo. A transmissão é baseada nos seguintes princípios:

a) cada segmento possui apenas uma fonte de alimentação;
b) quando a estação está mandando dados não existe energia no barramento;
c) todo equipamento possui um consumo constante de corrente;
d) são permitidas as topologias em linha, estrela ou árvore.

2.5.3 INTERBUS

O padrão Interbus foi concebido em 1984. Teve sua tecnologia desenvolvida pela empresa Phoenix Contact e utiliza o princípio mestre-escravo através do protocolo denominado "*One Total Frame*".

O sistema é composto por três tipos de elementos:

a) Controlador: o controlador é denominado *Host Controller Board* (HCB) e pode ser acoplado em PC’s ou CLP’s, ou ainda possuir interface para um nível mais alto de rede, desenvolvendo ao mesmo tempo funções de "*master*" e "*slave*";

b) Rede física: a rede física ou *remote bus* é constituída por um único cabo de comunicação, composto por três pares trançados. Pode-se utilizar também fibra óptica como meio físico. O cabo pode chegar até 13 km de comprimento, fazendo-se necessária a utilização de repetidores a cada 400m (metros);

c) Elementos de campo: os escravos (sensores e atuadores) podem ser inteligentes, não inteligentes ou mistos. Hoje estão disponíveis mais de 2000 produtos compatíveis, tais como: módulos de I/O remotos, terminais de válvulas, placas de interface para robôs, válvulas de controle, encoders, inversores de freqüência e outros.

Suas características são:

a) estrutura mestre-escravo (um único mestre);
b) possibilita transmissão em até 13km com restauração do sinal;
c) distancia máxima entre os nós de até 400 m;
d) escravos sem inteligência;
e) RS485 como meio físico;
f) taxa de 500 Kbits;
g) até 4096 nós podem ser conectados em anel;
h) popular especialmente na Europa.

2.5.4 *BITBUS*

O modelo de rede industrial *BitBus*, segundo Mondada ([MON2000]), tem que ter uma atenção especial, não por ser a melhor referência de rede industrial no mercado, mas por ter sido a primeira rede criada efetivamente para interagir com o campo. Foi utilizado primeiro em indústrias automobilísticas.

O *Bitbus* foi desenvolvido pela Intel no começo dos anos 80 como um sistema de comunicação aberto e suas especificações foram publicadas em 1983. Nessa época a Intel desenvolveu também um *chip* integrando as funções da proposta, facilitando o desenvolvimento de aplicações sobre o *Bitbus*.

Sua popularidade hoje ainda é muito grande, principalmente depois de 1991, onde uma comissão internacional baseou no *Bitbus* o padrão IEEE-1118. Suas características estão descritas abaixo.

a) estrutura mestre-escravo (um mestre; com a possibilidade de um novo mestre variável rapidamente);
b) mensagens de com até 248 bytes;
c) taxa de transmissão de 62,5 Kbits ou 375 Kbits;
d) distancia de até 13,5 km por segmento a 62,5 Kbits, ou 1200 m a 375 Kbits;
e) usa RS485 como meio físico, com par trançado, blindado preferivelmente;
f) possibilita *broadcasting* e *multicasting* do mestre. Mensagens enviadas para todos os escravos ou para um grupo seleto de escravos;
h) possibilita conectar 2 ou 3 repetidores em cascata;
i) popular mundialmente.

2.5.5 CAN

Este modelo foi desenvolvido pela *Bosh* em 1983 a 1985. Sua principal característica era a de reduzir a quantidade de fios que interligavam os equipamentos, centralizando o controle em uma unidade inteligente que iria distribuir as mensagens para o equipamento que interessasse. Com isso obtêm-se uma redução considerável no tráfego da rede e altas taxas de transmissão.

Porém, de início, teve-se problemas com as colisões dos pacotes na rede, onde dois nós respondiam simultaneamente. Esse problema foi solucionado colocando-se na *string* um número que significava a prioridade do pacote. Assim que houvesse colisão na rede, imediatamente a mensagem que possuía menor prioridade era parada, e a que possuía a maior prioridade era retransmitida. Outro fato é que os pacotes com volume de dados maiores possuem maior prioridade na rede.

Após isso, a *Bosh* juntou-se com a Intel para desenvolver um *chip* com as funções do padrão *CAN* para atuadores e dispositivos de entrada e saída. Atualmente existem várias propostas de padronização para comunicação em CAN, já que a tecnologia não trabalha com protocolo proprietário.

As características do modelo CAN são:

a) multi-mestre com acesso múltiplo;
b) taxa de transmissão de 20 Kbits, expansível até 1 Mbits. Em 40m a 1 Mbits e em 1 Km a 50 Kbits;
c) dificuldade de inserir repetidores;
d) meio físico RS485 com par trançado, preferencialmente brindado;
e) todas as mensagens tem um nível de prioridade;
f) há várias interfaces padrões de aplicação.

2.5.6 SERCOS

Sercos pertence a categoria de redes de campo dedicadas a uma aplicação específica. Sua primeira implementação foi desenvolvida em um instituto alemão em 1989, com o propósito de reduzir o tamanho de cabeamento para instalação de motores. Um número alto de nós pode ser conectados em sua topologia, que é em anel.

Os nós são conectados por fibra ótica, porque a comunicação entre os nós deve ser rápida. Uma série de parâmetros de comunicação também foi desenvolvida e já se encontra em um *chip* para ser usado em aplicações.

As principais características do Sercos são:

a) estrutura Mestre-Escravo;
b) topologia de Token Ring;
c) fibra Ótica como meio de transmissão;
d) até 254 escravos endereçáveis;
e) segmento com 60m para as fibras sintéticas;
f) taxa de transmissão de 2 a 4 Mbits.

3 PROTOCOLO *MODBUS*

Protocolo é, segundo Tanenbaum ([TAN1994]), um conjunto de regras e convenções para conversação. Essa regras definem a comunicação entre dois equipamentos, sejam eles computadores, máquinas ou computadores e máquinas. Nos protocolos são definidas as sintaxes como os equipamentos irão ordenar os dados de forma que fiquem entendidos por ambos os lados que fazem parte da comunicação.

O protocolo abordado nesse trabalho é o *Modbus* da companhia *Gold Modicon*. Este protocolo define uma estrutura de mensagem que os controladores reconhecerão e usarão, independente do tipo de rede acima deles. O protocolo *Modbus* também descreve o processo que um controlador usa para pedir acesso a outros dispositivos, como responderá a pedidos desses outros dispositivos, e como serão descobertos erros da comunicação e serão informados à sua origem. Em outras palavras, o protocolo fornece um formato comum para o plano e conteúdo de campos de mensagem ([MOR2000]).

Sendo o *Modbus* uma linguagem informática independente do material, esta permite o diálogo entre equipamentos de natureza e construtores diferentes. Também é importante ressaltar que existem várias implementações do protocolo *Modbus*. Como sua especificação é bastante ampla, as vezes não é necessário implementar todas as suas especificações para se ter uma rede industrial padrão *Modbus* em funcionamento, ou seja, tem-se como padrão de mercado o formato simples de mensagens que o *Modbus* utiliza.

Outra definição para o *Modbus* a nível de controladores, é que ele provê o padrão interno que os controladores usam para analisar gramaticalmente as mensagens. É o *Modbus*, que possibilita a um controlador que o mesmo reconheça uma mensagem se dirigida a ele, determine o tipo de ação a ser efetuada, e extraia os dados contidos na mensagem. Se uma resposta é requerida, o controlador construirá a mensagem de resposta e enviará a mesma usando o protocolo *Modbus* ([SCH2000]).

3.1 MODELO MESTRE-ESCRAVO

Em um rede modelo Mestre-Escravo, o protocolo define a comunicação, onde o mestre comanda a rede e os escravos, ouve a rede, e responde aos questionamentos quando

específicos a ele. Este tipo de rede, com mais de um escravo, é designado multiponto, que tem a vantagem de diminuir o cabeamento e o número de conexões que provem do mestre. ([MOR2000])

Na rede, o mestre pode questionar um escravo em particular e esperar pela sua resposta, o que chamamos de modo questão/resposta. Outra forma é o modo Difusão, onde o mestre manda um comando para todos os escravos da rede.

De modo geral, quando o mestre precisa de uma resposta do escravo, usa-se o modo questão resposta, e quando necessita-se programar todos os escravos, pode-se usar o modo difusão.

O modo difusão pode ser usado para questionar sobre um mesmo dado para todos os escravos somente quando houver uma comunicação também entre os escravos, transformando-os em mestre e escravos ao mesmo tempo. Um CLP ou computador pode então ser o administrador da rede no caso de rede mestre e escravos em cada nó, ou como mestre em redes onde um nó somente ouve e pergunta e os demais somente ouvem e respondem.

Geralmente, a troca de informações entre o mestre e o escravo se dá com a memória de dados. Porém, se a troca diz respeito à memória do programa, será necessário parar o modo questão/resposta antes de realizar a troca.

3.1.2 CICLO DE QUESTÃO-RESPOSTA DO MESTRE ESCRAVO (UM MESTRE E “N” ESCRAVOS)

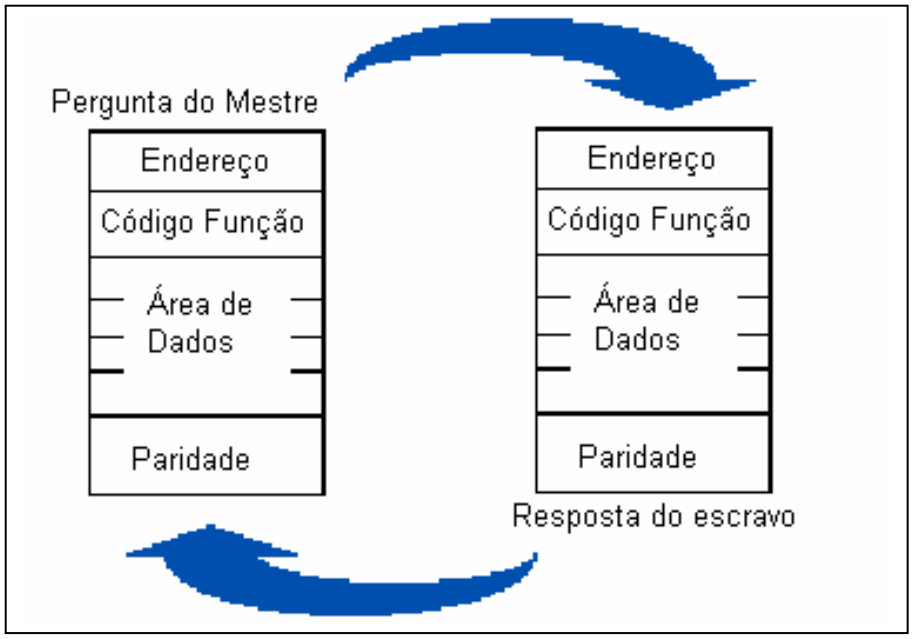
A PERGUNTA: O Código da função, significa a ação que o escravo irá receber e deverá executar. Os bytes de dados, contém informações adicionais que o escravo poderá precisar para executar a função. O campo de paridade provê um método para o escravo validar a integridade dos conteúdos da mensagem.

A RESPOSTA: Pode-se usar como código da função de resposta ao mestre, o mesmo código da pergunta, ou um outro código que o mestre reconheça. Os bytes de dados, contém as informações coletadas pelo escravo referente à pergunta do mestre. O campo de paridade, permite a validação da resposta pelo mestre.

Se a mensagem enviada ao escravo conter um erro, possibilitando mesmo assim que o escravo identifique que a mensagem é para ele, o mesmo retornará uma mensagem ao mestre, identificando a recepção errada das informações.

A figura 3.1, demonstra o ciclo de questão resposta no modelo mestre-escravo.

Figura 3.1 – Pergunta e resposta pelo protocolo *Modbus* ([SCH2000])



3.3 MODELO MESTRE-ESCRAVO ("N" MESTRES E "N" ESCRAVOS)

Nas redes com vários escravos e vários mestres, o funcionamento é parecido com o padrão *Token-Bus*, onde a máquina detentora do *Token*, define em que modo deseja transmitir na rede, se será no modo mestre ou no modo escravo. Se o mesmo encontra-se em modo mestre, o *Token* fica em seu poder até que ele receba a resposta do escravo. Após isso o *Token* é liberado na rede até que outro mestre se habilite ao mesmo.

3.4 MODELOS DE TRANSMISSÃO SERIAL COM *MODBUS*

Geralmente os controladores que utilizam o *Modbus* como protocolo de comunicação, podem ou não permitir a utilização dos dois modos de transmissão: ASCII (*American Standard Code for Information Interchange*) e/ou RTU (*Remote Terminal Unit*). Os usuários escolhem o modo desejado, juntamente com os parâmetros de comunicação como, taxa de transmissão, bits de paridade, etc.

A seleção de ASCII ou RTU definem o número de bits em um campo de mensagem transmitido serialmente na rede. Define também como serão empacotadas e decodificadas as informações.

3.4.1 MODO ASCII

Quando os controladores são organizados para comunicar em uma rede *Modbus* que usa o modo ASCII, cada um byte hexadecimal representa dois caracteres ASCII, ou seja, o número 05h é representado pelos caracteres 0 e 5 em ASCII, que em binário representam 0000000 e 00000101. A vantagem desse modo, é que permite um intervalo de até um segundo entre cada caracter enviado, sem causar um erro.

a) Sistema de Código

- Hexadecimal, ASCII caracteres: 0 .. 9, A .. F
- Um caracter Hexadecimal, contém dois caracteres ASCII

b) Sistema de transmissão

- 1 start bit;
- 7 data bits;
- 1 stop bit se a paridade é usada ou 2 stop bits se paridade não é usada

c) Campo de Paridade

- Paridade de Redundância Longitudinal (LCR)

3.4.2 MODO RTU

Quando os controladores são organizados para comunicar em rede *Modbus* que usa o modo RTU, cada byte, ou oito bits, representa dois números. Isso se deve ao fato do modo RTU representar valores dentro do padrão BCD Packed. O primeiro número é representado pelos quatro bits mais representativos e segundo pelos quatro bits menos representativos. Em resumo, teremos os números 1 e 9 representados por 00011001 que é o hexadecimal 19. A vantagem principal desse modo, é que sua maior densidade de caracteres, permite um melhor processamento dos dados que o modo ASCII em uma mesma taxa de transmissão.

a) Sistema de Código

- Hexadecimal: 0 .. 9, A .. F
- Dois caracteres Hexadecimal, contém oito bits do campo de mensagem

b) Sistema de transmissão

- 1 start bit;
- 8 data bits;
- 1 stop bit se a paridade é usada ou 2 stop bits se paridade não é usada

c) Campo de Paridade

- Paridade de Redundância Cíclica (CRC)

3.4.3 FORMATO DAS MENSAGENS NO *MODBUS*

No protocolo *Modbus*, as mensagens são começadas e terminadas por caracteres conhecidos tanto para os escravos como para os mestres. Esses caracteres não podem estar incluídos na lista de dados, pois quando o caracter de início de *string* é recebido, tanto o escravo, como o mestre, recebe serialmente os próximos caracteres a termina a recepção ao reconhecer o byte que simboliza o término da mensagem. Isso significa que se o escravo, por exemplo recebe no meio de uma mensagem um caracter que identifique um início de mensagem, ele irá tomar esse caracter como início de mensagem, causando um erro de CRC e retornando ao mestre.

3.4.3.1 FORMATO ASCII

No modo ASCII, toda mensagem começa com o caracter dois pontos (:), e finalizado por um caracter de *Carriage Return* (CRLF). Com o conhecimento desses caracteres, cada escravo fica esperando na rede um caracter de início de mensagem, e logo após o endereço do destinatário.

O intervalo para o envio e recepção de um byte, corresponde a um segundo, sendo que se o remetente não receber uma resposta, assumirá que algum erro ocorreu. O formato da mensagem do *Modbus* ASCII é mostrado na tabela 3.1.

Tabela 3.1 – Formato da mensagem *Modbus* em ASCII ([SCH2000])

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	N CHARS	2 CHARS	2 CHARS CRLF

3.4.3.2 FORMATO RTU

O padrão de mensagem no modo RTU é parecido com o modo ASCII, diferenciando-se por haver um tempo para o início e final da mensagem, em vez de caracteres especiais.

Os escravos ficam monitorando a rede, esperando por um tempo que varia de 3 a 5 bytes transmitidos de silêncio (T1-T2-T3-T4) tanto para início de mensagem, como para término da mesma. Assim que esse intervalo ocorrer, o dispositivo sabe que virá em seguida o endereço do destinatário e o restante da mensagem, até que o intervalo ocorrer novamente.

Com isso, se a mensagem não for transmitida de forma contínua, ou seja, se houver entre dois bytes transmitidos um intervalo grande de silêncio, o sistema irá detectar o próximo byte como sendo uma nova mensagem. O mesmo pode ocorrer se duas mensagem forem enviadas em tempos curtos de silêncio, fazendo com que o dispositivo receptor concatene as duas, causando um erro de CRC. O formato da mensagem em RTU é apresentado na tabela 3.2.

Tabela 3.2 – Formato da mensagem *Modbus* em RTU ([SCH2000])

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1.T2.T3.T4	8 BITS	8 BITS	N X 8 BITS	16 BITS	T1.T2.T3.T4

3.4.4 CONTROLE DO CAMPO DE ENDEREÇO

O campo de endereço, primeiro campo da mensagem, no modo ASCII possui dois bytes e no modo RTU um byte. Os endereços de dispositivos do protocolo *Modbus* é limitado a 247 dispositivos. Os dispositivos escravos são enumerados de 1 a 247. Um mestre dirige-se a um escravo, colocando o seu número no campo de endereço. Quando um escravo envia sua resposta, ele coloca o seu endereço no campo endereço para o mestre saber qual escravo está

respondendo, ou coloca o número do mestre na mensagem. O Endereço zero é usado no modo difusão.

3.4.5 CONTROLE DO CAMPO DE FUNÇÃO

O campo de função, que é o segundo campo da mensagem, no modo ASCII possui dois bytes e no modo RTU oito bits. As funções válidas comumente assumem os valores de 1 a 255 decimal.

O escravo ao receber uma mensagem proveniente do mestre, verifica no campo função, qual a ação que deve ser executada, e assim que executou a mesma, retorna uma mensagem, onde o campo de função pode ser o mesmo da pergunta, ou um código conhecido pelo mestre, que simbolize a execução correta.

Caso houve algum erro na transmissão dos dados, e o escravo conseguir detectar o erro, retornará para o mestre um código específico e conhecido por ambos, informando o erro.

3.4.6 CONTROLE DO CAMPO DE DADOS

O campo de dados, é constituído de um conjunto de dois dígitos binários no modelo RTU, ou um par de caracteres no modelo ASCII. Nesse campo contém informações adicionais, quando necessárias, nos comandos que trafegam via rede, tanto para o mestre como para o escravo. Essas informações servem por exemplo, para programar o escravo, receber dados do escravo, etc.

O campo de dados pode ser inexistente em certos comandos, como por exemplo, quando um mestre pede a um escravo os dados de produção. A função, por si só já faz com que o comando seja entendido pelo escravo, sem haver necessidade de dados adicionais.

3.4.7 CONTROLE DO CAMPO DE ERRO

Após o campo de dados, e antes da finalização da mensagem, é mandado, tanto para o escravo, como para o mestre, que faz a conferência da integridade dos dados enviados ou recebidos, um ou mais caracteres. É algo similar ao bit de paridade, na comunicação serial.

No modelo ASCII, o campo de erro pode conter um ou dois caracteres. Esses caracteres, são o resultado de um Cálculo de Redundância Longitudinal (LRC), cálculo que é executado com o conteúdo da mensagem, excluindo os caracteres de início e final de *string*. Após feito o cálculo, os caracteres de LRC são concatenados à mensagem como sendo o último campo antes do caracter de finalização.

No modelo RTU, o campo usado para receber o conteúdo do cálculo de integridade contém 16 bits, e o valor que esse campo irá receber, é o resultado de um Cálculo de Redundância Cíclica (CRC). Como no modelo ASCII, após feito o cálculo, os caracteres de CRC são concatenados à mensagem como sendo o último campo antes do caracter de finalização.

3.4.8 CÁLCULO DE PARIDADE

O cálculo da paridade no *Modbus*, possui formas distintas no modelo RTU e ASCII. Em testes detectou-se que o modelo ASCII possui menor confiança na paridade da mensagem, pois como trata-se de uma soma binária, a alteração da posição dos bytes dentro da mensagem, não interfere no cálculo do LRC, porém, torna a mensagem inválida. Já no cálculo do CRC do modelo RTU, qualquer troca de posição implica na não validade da paridade.

No protótipo, seguiu-se o modelo ASCII tanto para transmissão dos caracteres, como para cálculo da paridade..

4 MEIO DE TRANSMISSÃO RS485 – RS232C

As redes industriais, conforme já foi comentado, utiliza-se dos padrões seriais para transmissão dos dados. Abaixo descreve-se a utilização e compatibilidade de comunicação entre uma interface RS485 (utilizada em redes de campo) e RS232C, padrão este, que vem incorporado na maioria dos PC´s atualmente no mercado.

O RS485 possui transmissão balanceada e suporta conexões multi-ponto (*multidrop*), o que permite a criação de redes com até 32 nós e transmissão à distância de até 1200 metros ([FAR1998]).

Segundo Klitzke ([KLI1999]), “quando é necessário a transmissão de pequenos blocos de dados por longas distâncias, a RS485 é a melhor escolha”. Sendo uma de suas características a utilização de duas linhas balanceadas, a RS485 se torna mais imune a ruídos elétricos encontrados em ambientes industriais.

Outra característica do RS485 é, ao contrário da RS232C que utiliza níveis de voltagem para representar os bits 0 e 1, a utilização de níveis de corrente para representar os bits 0 e 1. É essa característica que permite a RS485, a transmissão a longas distâncias comparando-se a distância alcançada pela RS232C.

Segundo Franco ([FRA2000]), “erroneamente tem-se o conceito de que estes padrões definem protocolos de comunicação específicos. Os padrões ANSI/EIA RS-xxx especificam apenas as características elétricas.” Deve-se lembrar também, que o padrão RS485 não é por si só uma tecnologia de rede industrial, mas pode ser usado para o mesmo.

4.1 DESCRIÇÃO DO BARRAMENTO RS485

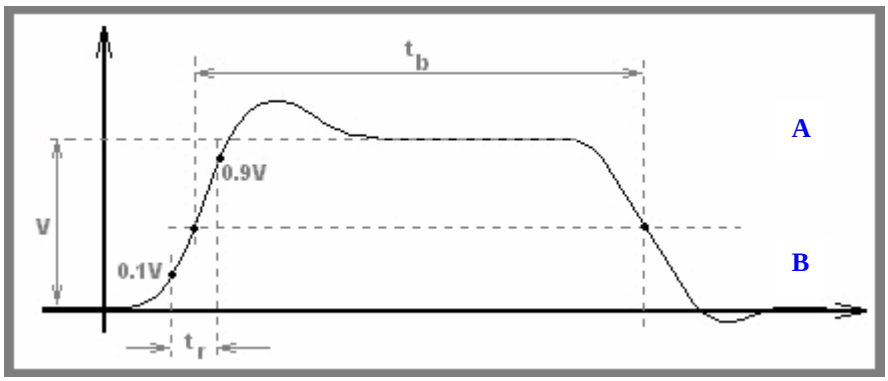
A lógica para transmissão de um byte no padrão RS485 é definida quando um terminal torna-se mais negativo ou mais positivo que o outro. Portando, a tensão diferencial entre os dois terminais permitirá o reconhecimento do bit que está sendo transmitido, se é "0" ou se "1".

Convencionalmente, a lógica "1" é reconhecida quando o terminal A do transmissor torna-se mais negativo em relação ao terminal B. A lógica "0" é identificada quando o terminal A torna-se positivo em relação ao terminal B.

É por esse motivo que tem-se maior imunidade a ruídos eletromagnéticos externos. O gerador também deve ser capaz de limitar a corrente em 150mA no caso de um curto-circuito para o terra. Para o receptor, uma tensão diferencial de entrada de 200mV é o mínimo suficiente para que se possa identificar uma lógica, operando normalmente em tensões

diferencias de até 6V. Porém o equipamento deve suportar até 12V de tensão sem que seja danificado. A figura 4.1, demonstra o sinal no barramento RS485.

Figura 4.1 – Transmissão no barramento RS485 ([FRA2000])



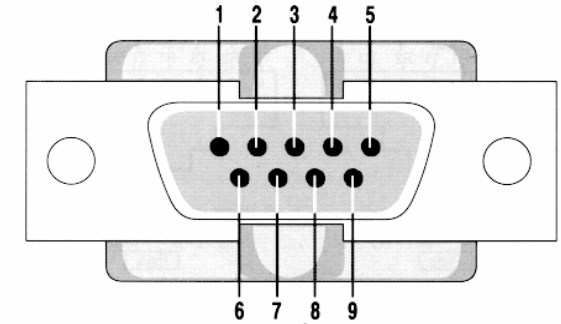
V - diferença entre os níveis de tensão

t_r - tempo de transição entre 10% de V e 90% de V

t_b - largura do pulso entre os níveis 0.5V sucessivos

4.2 DESCRIÇÃO DO BARRAMENTO RS232C

O padrão RS232C pode ser implementada tanto com 9 sinais (conforme figura 4.2), o qual conhecemos pelo conector BD9, e sua designação normativa como ComPort EIA-574, como em um conector 25 pinos.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

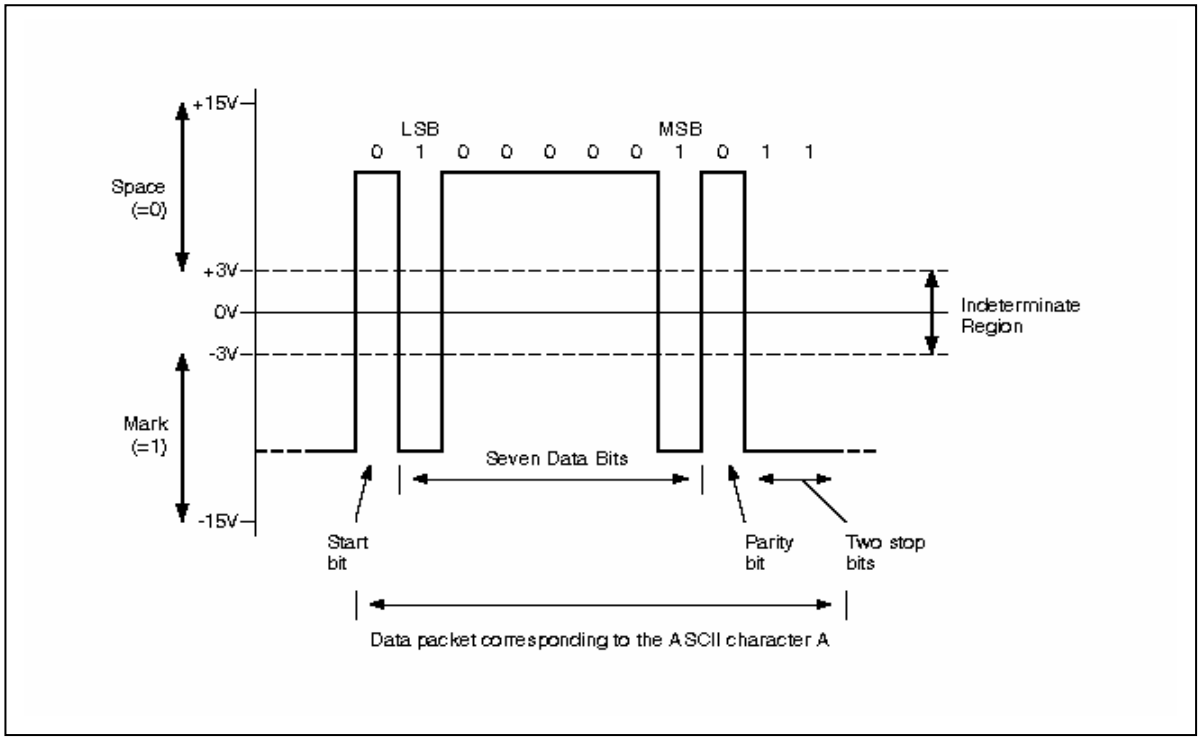
Figura 4.2 - Pinagem do conector RS232C DB9 ([ARC2000])

Para ser possível a transmissão de dados na RS232C, é necessário haver uma seqüência de inicializações do meio. A inicialização basicamente corresponde ao protocolo descrito abaixo, onde de um lado temos a porta serial RS232C chamada aqui de controladora, e do outro lado temos o dispositivo receptor, chamado aqui de portadora ([OLI1987]):

a) Controladora OK = A controladora envia o sinal de DTR para a portadora (Pino 4);
b) Portadora OK = A portadora envia o sinal de DCD para a controladora (Pino 1);
c) Portadora Pronta = A portadora envia o sinal de DSR para a controladora (Pino 6);
d) Permissão p/ TX = A controladora envia o sinal de RTS para a portadora (Pino 7);
e) Permissão OK = A portadora envia um sinal de CTS para a controladora (Pino 8);
f) Transmissão = Intercala-se TX e RX entre controladora e portadora (Pinos 3 e 2);

Na figura 4.3, é apresentada a forma de transmissão elétrica de um caracter (7 bits) no RS232C.

Figura 4.3 – Transmissão de um byte Assíncrono ([ARC2000])



4.3 INTERFACES RS485 PARA RS232C E VICE-VERSA

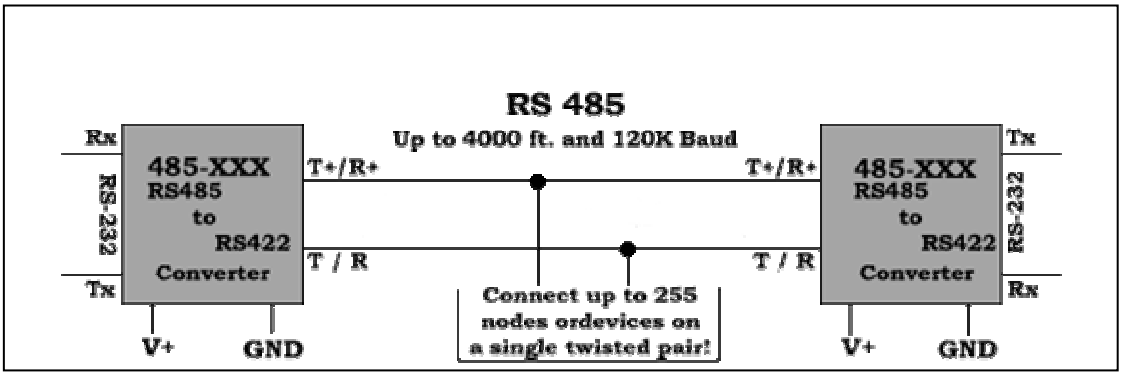
Sendo a RS232C a implementação padrão para comunicação serial nos PC´s, tem-se que fazer a conversão dos sinais do padrão RS232C para RS485 e vice-versa para que haja a comunicação correta entre os equipamentos que trabalham em uma rede industrial RS485 e o PC.

Existem hoje no mercado vários conversores de RS, que praticamente compatibilizam a comunicação serial entre os vários padrões.

O padrão serial RS232C, sendo desenvolvido para rede ponto-a-ponto, é orientado a conexão, ou seja, deve haver um dispositivo no outro lado da linha, pronto para receber e enviar os dados segundo o seu padrão. É nesse ponto que o conversor de RS opera. Suas principais características são, em se tratando de um conversor de RS485 para RS232C, a simulação de um dispositivo que responda aos questionamentos da RS232C para efetuar a conexão e a passagem dos dados para a RS485. Como a RS485 não é um padrão orientado a conexão, a qualquer hora, qualquer equipamento poderá transmitir seus dados na rede. Aí é

que entra a figura do mestre da rede, como já vista anteriormente, para coordenar esse tráfego. Na figura 4.4 temos um exemplo do funcionamento da compatibilidade de RS232C e RS485.

Figura 4.4 – Exemplo de conexão RS232C – RS485 ([SOA1993])



4.4 ESPECIFICAÇÃO TÉCNICA DOS PADRÕES SERIAIS

Na Tabela 4.1 é apresentado a especificação e algumas características dos padrões seriais mais conhecidos e utilizados do mercado.

TABELA 4.1 - ESPECIFICAÇÃO TÉCNICA DOS PADRÕES SERIAIS

SPECIFICATIONS	RS232C	RS423	RS422	RS485
Mode of Operation	Single Ended	Single Ended	Diffe-rential	Diffe-rential
Number of Drivers and Receivers	01 Driver 1 Receivers	01 Driver 10 Receivers	01 Driver 10 Receivers	01 Driver 32 Receivers
Maximum Cable Length	50 Ft.	4000 Ft.	4000 Ft.	4000 Ft.
Maximum Data Rate	20 kb/s	100 kb/s	10 Mb/s	10 Mb/s
Maximum Driver Output Voltage	+/-25V	+/-6V	-0.25V to +6V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	+/-5Vto +/-15V	+/-3.6V	+/-2.0V	+/-1.5V
Driver Output Signal Level (Unloaded Max)	+/-25V	+/-6V	+/-6V	+/-6V
Driver Load Impedance (Ohms)	3k to 7k	>=450	100	54
Max. Driver Current in High Z State Power On	N/A	N/A	N/A	+/-100uA
Max. Driver Current in High Z State Power Off	+/-6mA @ +/-2v	+/-100uA	+/-100uA	+/-100uA
Slew Rate (Max.)	30V/uS	Adjustable	N/A	N/A
Receiver Input Voltage Range	+/-15V	+/-12V	-10V to +10V	-7V to +12V
Receiver Input Sensitivity	+/-3V	+/-200mV	+/-200mV	+/-200mV

Receiver Input Resistance (Ohms)	3k to 7k	4k min.	4k min.	>=12k

4.5 API DO WINDOWS PARA ACESSO A PORTA SERIAL

Segundo Longo ([LON1997]), um recurso de comunicação para o Sistema Operacional Windows, é um dispositivo físico ou lógico que proporciona um fluxo de dados assíncrono e bidirecional. Portas seriais, paralelas, máquinas de fax e modems são exemplos desses recursos. Para cada recurso existe um provedor de serviços, também chamado de biblioteca. Essas bibliotecas disponibilizadas pelo Windows, constituem as conhecidas funções de API (*Application Program Interface*), que são funções que o sistema operacional coloca a disposição dos desenvolvedores para acessos aos diversos dispositivos do sistema.

Essas funções de API são geralmente implementadas em *DLLs (Dynamic Link Library*) que são os arquivos em que as funções estão contidas. Em termos de comunicação serial, a operação básica via API constitui em abrir uma porta serial, configurar a velocidade e tempos, após isso iniciar a transmissão e recepção dos dados, colocando a porta em modo de TX (Transmissão) ou RX (Recepção) ([DAV1997]).

Enviar dados por uma porta serial, é na verdade escrever byte a byte, todo o conteúdo de uma informação em um endereço de entrada e saída (I/O) ([LON1997]). Porém, a porta serial não consegue transmitir dados na mesma velocidade que se consegue suprir os dados para ela. Para contornar esse problema, a porta serial levanta um *flag* sinalizando que está ocupada ou livre para transmissão (TX) ou recepção (RX) .

Nos anexo 2 encontra-se as funções de manipulação da API do Windows para acesso a porta serial.

5 AMBIENTE DE TRABALHO

Abaixo será descrito o ambiente de trabalho onde se trabalhou, bem como uma descrição técnica e funcional dos equipamentos de foram necessários no decorrer do desenvolvimento do protótipo.

5.1 MÁQUINA DE EMPACOTAR

As máquinas de empacotar ou empacotadoras, são equipamentos para o empacotamento de produtos que são utilizados no dia-a-dia dos consumidores. Utiliza-se as mesmas para açúcar, trigo, arroz, farinha, biscoitos. Enfim, na maioria dos produtos que se encontra nas prateleiras dos supermercados, a probabilidade do mesmo ter sido empacotado por um processo automatizado (empacotadora) é grande. As máquinas de empacotar existentes no mercado nacional, são basicamente das marcas Indumak, Bosh, Halmak entre outras.

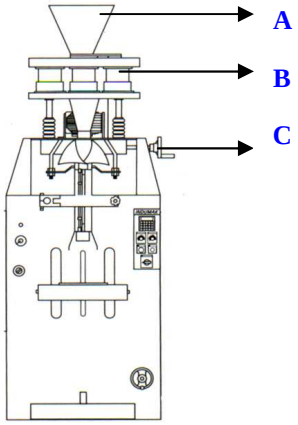
O funcionamento das mesmas é basicamente igual, diferenciando-se apenas por detalhes como consumo de energia e alguns movimentos mecânicos. O empacotamento do produto, é um dos processos finais de todo o processo produtivo de uma empresa alimentícia. Empresas do ramo açucareiro e cerealista são as que mais utilizam as máquinas de empacotar, pois o seu produto precisa estar envólucro em uma embalagem até chegar ao consumidor final.

Como exemplo de processo, cita-se o açúcar, que após passar por todos os processos produtivos, desde a moagem da cana-de-açúcar até o refinamento, é levado por dutos até um setor da empresa onde estão instaladas as máquinas de empacotamento. Cada máquina possui, conforme demonstra a figura 5.1 e 5.2, um pequeno reservatório na parte superior, onde fica armazenado uma quantidade para ser empacotada. Essa quantidade varia, mas para se ter um parâmetro, cita-se 15 kg. As máquinas empacotam desde gramas até pacotes de 5kg. Em relação ao açúcar, cita-se 1kg, 2kg e 5kg,

Logo abaixo desse reservatório encontra-se as canecas, em números de 2, 3 ou 4 canecas. As mesmas tem a função de girarem em volta de um eixo, e sobre um plano horizontal. Na parte frontal da máquina, esse plano possui um orifício, sobre o qual, a caneca passa e deixa cair o seu conteúdo. Esse conteúdo (o produto) desliza por um tudo de aproximadamente 1,5 m (varia de acordo com o fabricante), onde na outra extremidade encontra-se um pacote vazio e fechado na parte superior, para receber o produto.

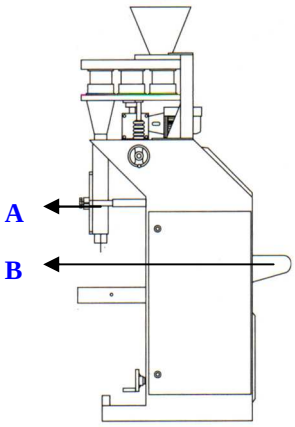
Com isso se houver canecas de 1Kg de volume, a cada passada de uma caneca pelo orifício da base sobre a qual se encontra, encher-se-á um pacote de açúcar de 1Kg. Máquinas que empacotam 2 Kg, possuem a caneca maior, ou são adaptações feitas nas máquinas de 1Kg onde, faz-se com que a mesma deixe passar duas canecas de 1Kg antes de fechar o pacote. Na figura 5.1 e figura 5.2, é ilustrada uma máquina de empacotar.

Figura 5.1– Máquina de Empacotar
Visão Frontal



A) reservatório;
B) canecas;
C) manipulo de correção.

Figura 5.2– Máquina de Empacotar
Visão Lateral



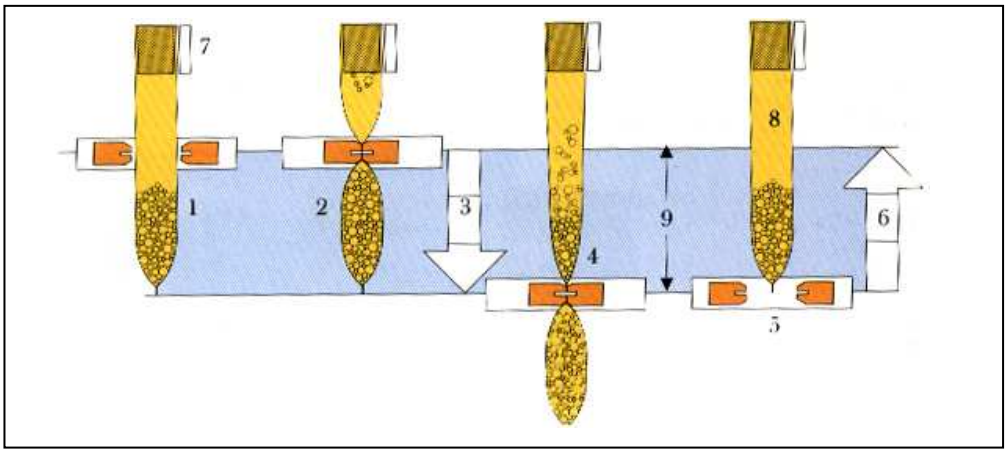
A) tubo para passagem do produto;
B) bobina plástica.

Em relação ao fechamento do pacote, e utilização do filme plástico, a máquina trabalha da seguinte forma. Na parte de trás da mesma, é colocado uma bobina com o filme plástico aberto. O filme, é passado para a frente da máquina, envolvendo o tubo que transporta o produto. Esse filme possui de acordo com o comprimento da embalagem uma

marca escura, a qual passa por um sensor (foto célula) e identifica o fim de um pacote e início de outro. Esse sensor é o responsável pelo acionamento das canecas e corte do filme.

O corte do filme plástico é feito por um mecanismo chamado de mesa. A qual após o pacote estar cheio, corta o mesmo, fecha a parte superior do pacote atual e a parte inferior do próximo pacote. A mesa possui um movimento de cima para baixo e abre e fecha, de acordo com a figura 4.3. No ponto 1 a mesa está aberta e na parte superior de sua trajetória vertical. Após o pacote ser enchido, a mesa se fecha, e executa a função de cortar o pacote e fechar a parte superior do atual e a parte inferior do próximo. Na trajetória do ponto 2 para o ponto 4 (descida), a mesa puxa o próximo pacote para baixo, esperando a caneca despejar o produto. Ao chegar novamente ao ponto 5, ela se abre e começa o caminho novamente ao ponto 1 (movimento de subida), voltando todo o ciclo. Conforme mostra a figura 5.3:

Figura 5.3 - Ciclo do fechamento do pacote



Um sincronismo entre as canecas, o foto sensor do filme plástico e a mesa, faz com que os produtos sejam envasados e após o corte e fechamento do pacote o mesmo segue para uma esteira ou um controlador de peso.

5.2 CONTROLADOR DE PESO

Como já foi visto o funcionamento e a funcionalidade de uma máquina de empacotar, fala-se agora sobre controladores de peso. Não necessariamente toda máquina de empacotar possui um controlador de peso. São equipamentos totalmente diferentes, porém completam juntos um ciclo do processo produtivo das empresas que necessitam empacotar seus produtos.

Até 1988 aproximadamente, os pacotes produzidos por uma máquina de empacotar eram colocados diretamente nas esteiras, enfardados e despachados para o consumidor final ([LON1997]). Como as máquinas de empacotar são produzidas de acordo com o produto que irão trabalhar, cada caneca é adaptada para conter exatamente o peso correspondente a sua especificação ([URB1993]). Ou seja, uma máquina que empacota açúcar de 1kg, pode ter canecas diferentes da mesma máquina que empacota arroz de 1kg. Essa diferença deve-se ao fato da diferença de volume e densidade dos produtos, causando a diferença também no peso ([LON1997]). É como aquela velha estória: “o que pesa mais? 1kg de pena ou 1kg de ferro? ” Apesar do peso ser o mesmo, os volumes são bem diferentes.

Porém, verificou-se também, que um mesmo produto, pode variar de volume durante o processo, causando uma variação no peso. No caso do açúcar, por exemplo, a umidade causa um aumento considerável do volume, mas não do peso. Com isso as máquinas de empacotar, produziam pacotes dentro da sua especificação original e também abaixo ou acima da mesma. Essa variação tornou-se uma forma de ganho para as empresas e perda para o consumidor, ou vice e versa.

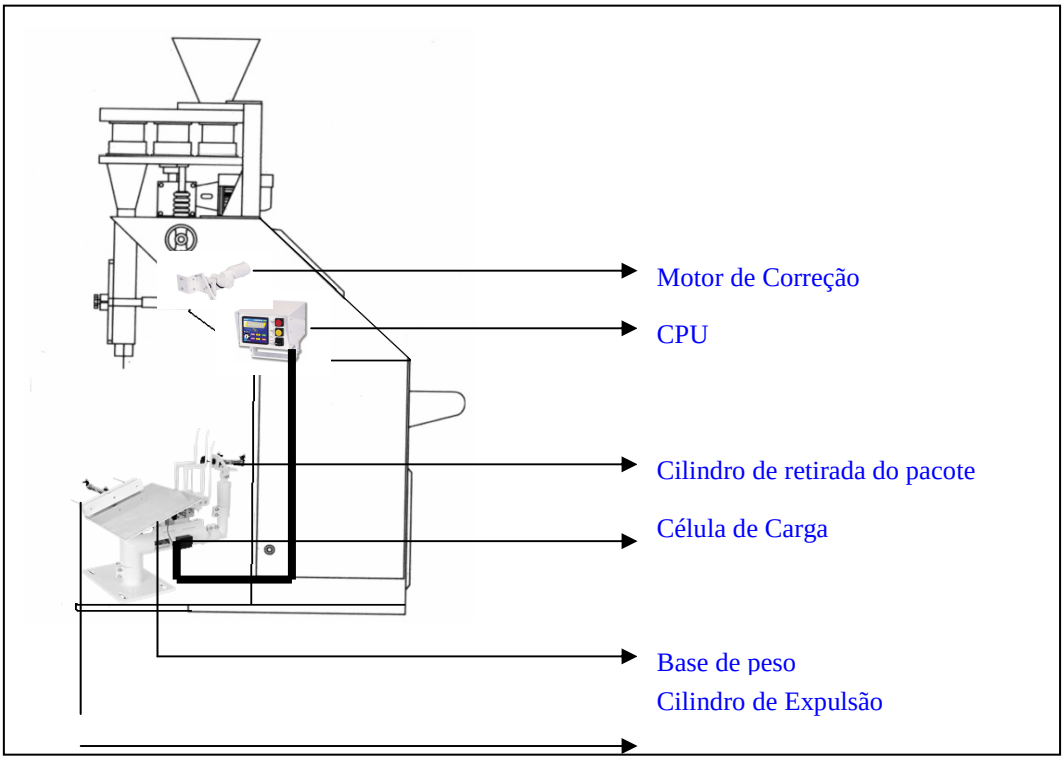
As máquinas de empacotar, possuem um dispositivo, chamado de manípulo de correção, para aliviar esse problema. Trata-se de uma roda que movimenta-se para a esquerda e direita, aumentando ou diminuindo o volume das canecas. Porém esse processo até então era manual. O operador da máquina de empacotar, possuía uma balança ao lado, onde periodicamente ele pegava um pacote produzido, e verificava o peso do mesmo. Se estivesse fora dos padrões, manualmente girava a roda, corrigindo o peso para mais ou menos volume nas canecas, sem parâmetros estatísticos para tal. Tudo se dava pelo conhecimento do operador e sua experiência.

Outro fato, que ajudou a disseminação dos controladores de peso na indústria alimentícia, foi a implantação da portaria 74 pelo órgão governamental INMETRO (Instituto de Metrologia). Tal órgão, tem a responsabilidade de defender os interesses do consumidor em relação a qualidade dos produtos que compram. Um item que diz respeito a essa qualidade está nas medidas e pesos corretos nominais. A portaria 74, diz respeito ao peso dos produtos.

Se uma empresa estiver praticando ganhos em relação ao peso descrito nas embalagens dos seus produtos, a mesma será multada, de início, e poderá após reincidência dessas falhas, perder até a concessão para produção do produto ([INM1992]).

Foi nesse contexto que surgiram os controladores de peso, com a proposta de manter o peso correto nas embalagens mesmo com a constante variação do produto. O controlador é instalado abaixo do tubo das empacotadoras, fazendo com que o pacote, após ser cortado, caia em cima de uma bandeja, que faz parte do controlador. Abaixo temos a figura 5.4, que demonstra a instalação do controlador de peso na máquina de empacotar.

Figura 5.4 – Controlador de peso instalado na máquina de empacotar

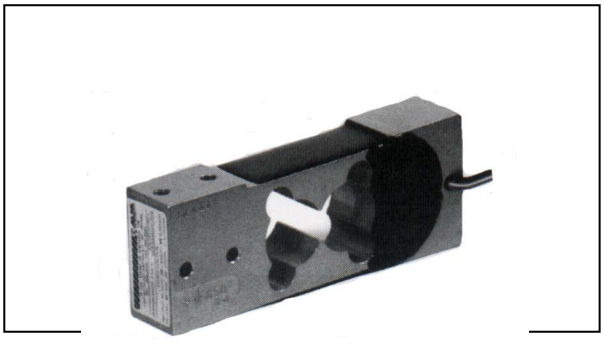


Os componentes básicos de um controlador de peso são a célula de carga, o módulo de controle, parte pneumática e motor de correção. Todos serão descritos de forma detalhada abaixo.

5.2.1 CÉLULA DE CARGA

Dispositivo desenvolvido para transformar uma pressão sobre sua superfície em peso. É basicamente um retângulo com um circuito, que transfere sinais analógicos para o módulo de controle que os transforma em dados digitais, representando o peso. É sobre a célula de carga que fica a base de peso, que recebe o pacote da empacotadora. Assim que o pacote cai sobre a célula de carga, a mesma envia sinais analógicos para o módulo central de controle. Ao cair na bandeja sobre a célula de carga, o pacote causa uma vibração. Somente após a vibração parar consideravelmente, é que o pacote poderá ser pesado e verificado se o mesmo está dentro dos padrões ou não ([URB1993]). Tal vibração, é uma das causas responsáveis pela velocidade de trabalho dos controladores de peso. Ela dura em média 300ms para estagnar, o processo de peso dura mais 300ms e a retirada do pacote mais 300ms; considerando uma máquina de 1Kg. Em média, costuma-se trabalhar com controladores pesando de 20 ppm (pacotes por minuto) a 130 ppm. Outro ponto relevante à velocidade dos controladores, é a própria máquina de empacotar. Nos padrões de mercado, as máquinas que empacotam 5 Kg, trabalham com velocidade média de 30 ppm, as máquinas de 1 Kg tem velocidade média de 60 ppm. Na figura 5.5 mostra-se a Célula de carga da Alfa, um fabricante nacional.

FIGURA 5.5 – CÉLULA DE CARGA

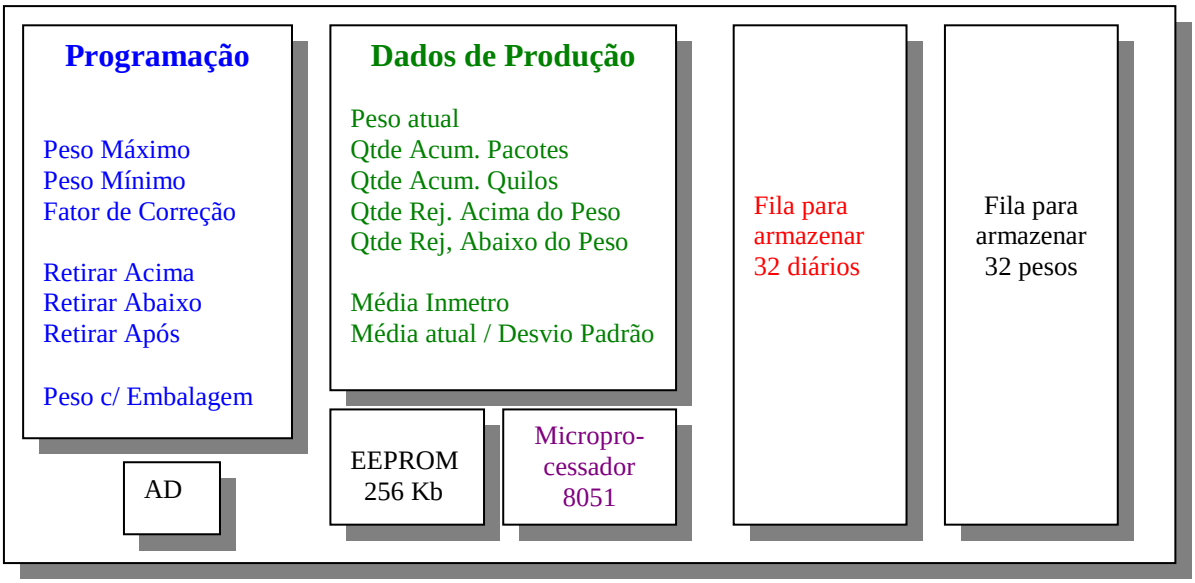


5.2.2 MÓDULO DE CONTROLE

É a parte do controlador de peso que recebe os dados analógicos da célula de carga, transforma-os em digitais, via um AD. De acordo com os dados faz-se todo o

processamento necessário. Um esquema simplificado do controlador de peso é mostrado no quadro 5.1:

1.1.1.1.1.1.1.1 QUADRO 5.1 – ESQUEMA SIMPLIFICADO INTERNO DO CONTROLADOR DE PESO



Ao receber o dado analógico de peso, a CPU o transforma em digital e verifica o valor do mesmo. O peso de referência, ou seja, o peso ideal, é a média do peso máximo e o peso mínimo, gravado em memória do controlador. Por tanto, se o peso atual, for igual ao peso médio calculado, o controlador coloca o peso na fila de pesos, recalcula os acumuladores de produção e médias, e aciona a parte pneumática do controlador para retirar o pacote da base de peso. Já se o pacote não for igual ao padrão, verifica-se e o mesmo está dentro dos pesos aceitáveis (entre retirar acima e retirar abaixo). Se o mesmo não estiver, aciona-se a parte pneumática responsável pela retirada do pacote da bandeja, e também a parte pneumática responsável pela retirada do pacote da esteira. O mesmo é jogado em um recipiente ao lado da esteira, para reprocesso. Na figura 5.6, apresenta-se a CPU do controlador de peso.

FIGURA 5.6 – CPU DO CONTROLADOR DE PESO DWA CHECK 5P

Após a verificação de um pacote diferente do padrão, o controlador aciona a saída responsável pela correção das canecas da máquina de empacotar. É o inter-relacionamento entre o controlador de peso e a máquina empacotar. Ligado a uma das saídas do controlador de peso, está o motor de correção, visto na figura 5.6, responsável pela correção automática das máquinas de empacotar. Essa correção resume-se em um motor de redução, responsável em girar o manipulo de correção da máquina de empacotar para direita ou esquerda, aumentando ou diminuindo o volume das canecas de dosagem. Tal processo leva sempre em conta o peso médio. Supondo que o peso médio desejado, em uma máquina de 1kg seja de 1006,0 gramas, e o peso médio atual (média aritmética dos últimos 32 pacotes armazenados) seja de 1004,0, o motor irá corrigir as canecas para mais volume. Se o peso médio ultrapassar a média desejada, o mesmo irá diminuir o volume, mantendo-se sempre uma aproximação da média. Com isso teremos em todo o processo produtivo, pesos abaixo da média e acima, equilibrando a média da produção total. Todo esse processo é baseado nos dados de programação, ou seja, se o controlador for programado errado, o controle também sairá errado. Abaixo segue a explicação mais detalhada sobre a programação do controlador.

5.2.2.1 ESPECIFICAÇÃO DA PROGRAMAÇÃO DO CONTROLADOR DE PESO

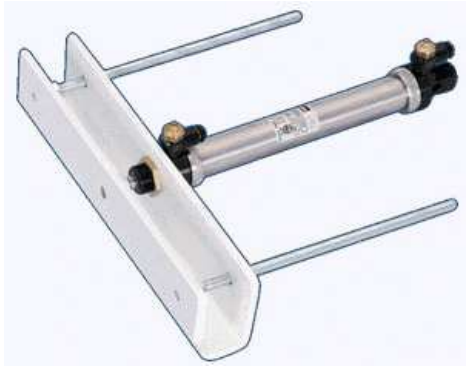
a) peso máximo: parâmetro para se efetuar a correção do peso para cima;
b) peso mínimo: parâmetro para se efetuar a correção do peso para baixo;

c) fator de correção: é o tempo que o motor de correção deverá ficar acionado para corrigir;
d) retirar acima: parâmetro para retirada do pacote da esteira e reprocesso do mesmo;
e) retirar abaixo: parâmetro para retirada do pacote da esteira e reprocesso do mesmo;
f) retirar após: tempo que o sistema deverá esperar para acionar a parte pneumática responsável pela retirada do pacote com peso fora dos padrões;
g) peso com embalagem: peso ideal do sistema.

5.2.3 PNEUMÁTICA

A parte pneumática do controlador de peso, é composta basicamente de dois cilindros, responsáveis um, pela retirada do pacote da bandeja de pesagem, e outro pela retirada do pacote da esteira, caso o mesmo esteja totalmente fora dos padrões programados no controlador. Na figura 5.7 é apresentado o cilindro de expulsão.

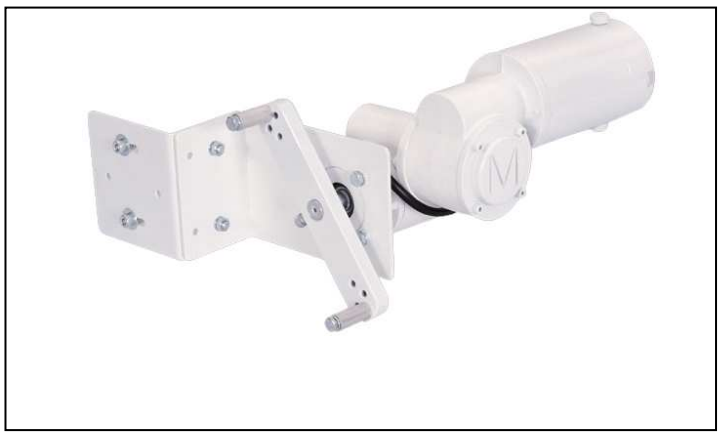
FIGURA 5.7 – CILINDRO PNEUMÁTICO QUE COMPÕE O CONTROLADOR DE PESO



5.2.4 MOTOR DE CORREÇÃO

Como já mencionado, o motor de correção (figura 5.8), é responsável pelo aumento ou diminuição do volume nas canecas da empacotadora. Com esse processo pode-se corrigir, gradativamente, as variações do volume do produto.

FIGURA 5.8 – MOTOR DE CORREÇÃO



Com isso verificamos a importância do controlador de peso, seja para manter a menor variação entre os pacotes envasados em relação do peso médio desejado, evitando assim multas do INMETRO e o desperdício do processo produtivo.

6 DESENVOLVIMENTO DO PROTÓTIPO

O desenvolvimento do protótipo utilizando RS485 e *Modbus* como protocolo de comunicação em um PC, passa obrigatoriamente, pela interface serial do microcomputador,

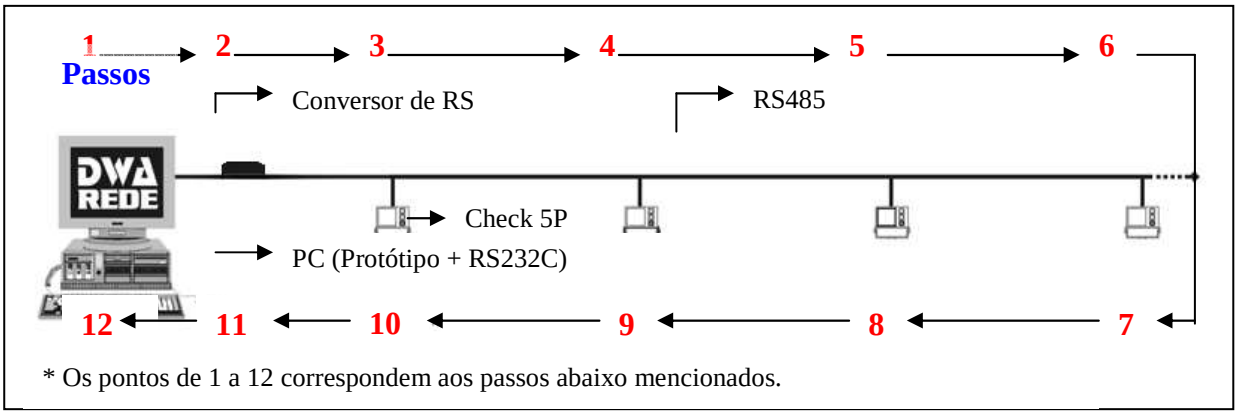
ou seja, a RS232C. A nível de especificação e implementação do protótipo na arquitetura IBM PC, deve-se dar uma atenção especial a programação da interface serial, ou seja, configurá-la de forma correta. Outro ponto importante é a montagem das mensagens que serão transmitidas do PC para o controlador de peso e vice-versa dentro das especificações do *Modbus*. Como a RS485 é somente o meio que utilizar-se-á para o tráfego das informações, a única atenção em relação a ele deve ser na montagem do cabeamento.

Para a especificação do protótipo, foi utilizada a fluxogramação, que segundo Klitzke ([KLI1999]), para processos com execução seqüencial, o fluxograma é um método de especificação e documentação bastante conhecido e utilizado. Em concordância com o autor, utilizou-se esse método para especificação do protótipo. Para a metodologia de desenvolvimento do protótipo será utilizada Análise Estruturada. O aplicativo Micrografx ABC FlowCharter, em sua versão 4.0, mostrou-se uma ferramenta muito útil para se aplicar a técnica de fluxogramação na especificação do protótipo.

6.1 ESPECIFICAÇÃO DO PROTÓTIPO

Para se ter um entendimento melhor sobre a especificação e operacionalidade do protótipo, é necessário ver primeiramente a instalação física da rede de controladores de peso e uma visão geral do processo. Basicamente o processo de busca e envio de informações vai ocorrer como demonstra a figura 6.1 abaixo:

Figura 6.1 – Diagrama macro do ambiente



A seguir são descritos os passos acima relacionados.

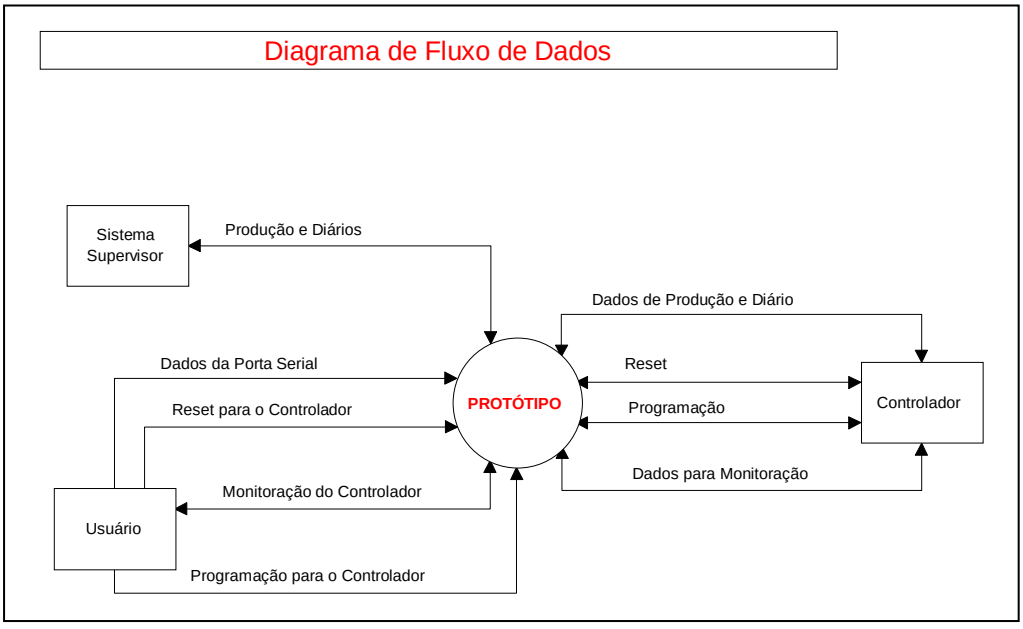
6.1.1 DESCRIÇÃO DOS PASSOS

Partindo-se do passo 1 tem-se o computador, no qual se encontra o protótipo e a porta serial RS232C. A mensagem do protótipo (mestre), é enviada para a RS232C e depois passa para o passo 2, onde encontra-se o conversor RS232C para RS485. Após o conversor, a mensagem passa para o meio RS485 e chega até os controladores de peso (escravos), como segue os passos de 3 à 6, que irão identificar para quem é a mensagem. O controlador questionado irá responder, passando a resposta para todos os controladores novamente (passos de 7 à 10), até chegar no conversor (passo 11) que irá converter de RS485 para RS232C. A mensagem é recebida pelo PC na porta serial (passo 12) e por fim, o protótipo lê a porta serial obtendo a resposta em questão.

6.1.2 DIAGRAMA DE FLUXO DE DADOS (DFD)

O diagrama de contexto abaixo exibe as entradas e saídas do protótipo.

Figura 6.2 – Diagrama de Fluxo de Dados (DFD)



No diagrama de fluxo de dados pode-se observar as principais funções do protótipo. Onde um sistema de terceiros irá pedir a cada minuto as informações de produção e diários. A nível de usuário, o mesmo poderá interagir com o sistema, configurando os parâmetros seriais

e utilizando as ferramentas de monitoração, alteração de parâmetros e *reset* dos controladores. As informações enviadas ao controlador, todas, sem exceções terão um retorno ao protótipo, significando um comando foi executado com sucesso ou não, ou o recebimento de informações do processo de peso.

6.1.3 FLUXOGRAMA DA CHAMADA DAS FUNÇÕES

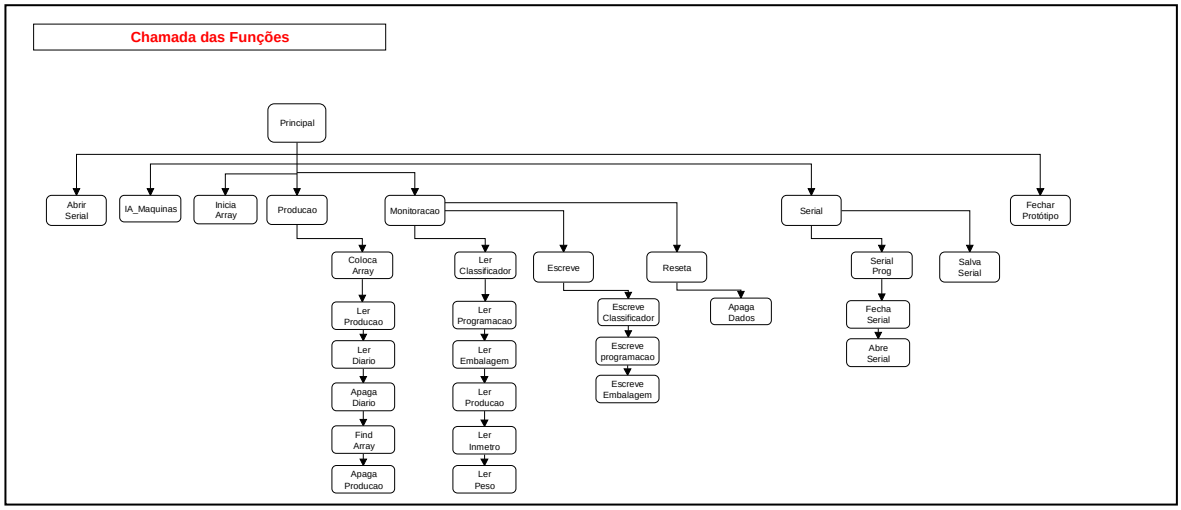
Na figura 6.3, é apresentada a ordem das chamadas das funções e também o nível que cada função está especificada no protótipo.

Como funções de leitura tem-se: Lerpeso, Lerproducao, Lerprogramacao, Lerclassificador, Lerdiaio, Lerpeso e Lerinmetro.

Como funções de escrita no controlador tem-se: Escreverprogramacao, Escreverclassificador, Escreverembalagem.

Como funções de eliminação de dados tem-se: Apagarproducao, Apagarpilha, Apagardiario.

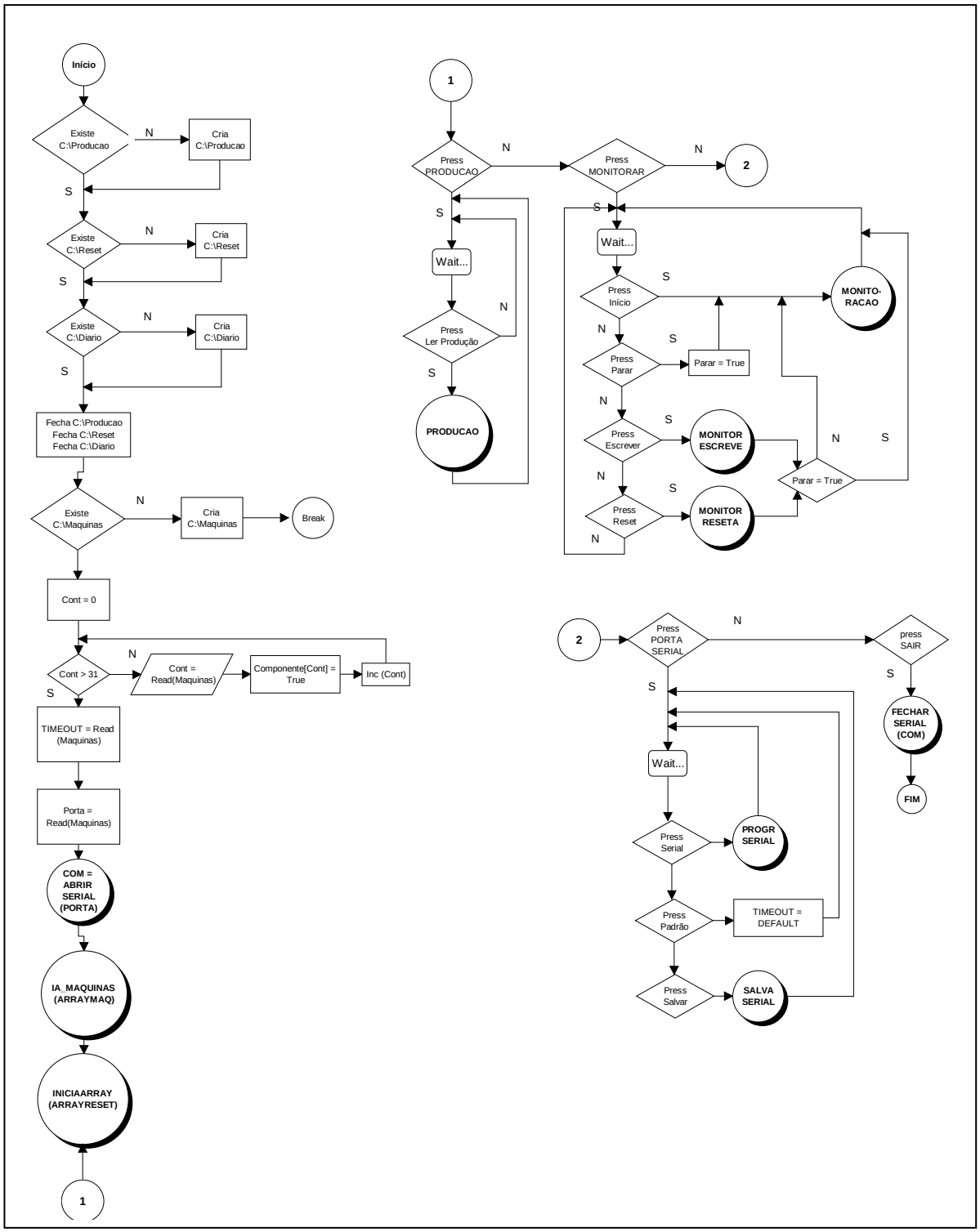
Figura 6.3 – Chamada das funções no protótipo



Como pode-se observar na figura 6.3 o protótipo primeiro abre a porta serial, inicializa um *array* de máquinas ativas no sistema e um *array* de máquinas para *reset*. Após esse

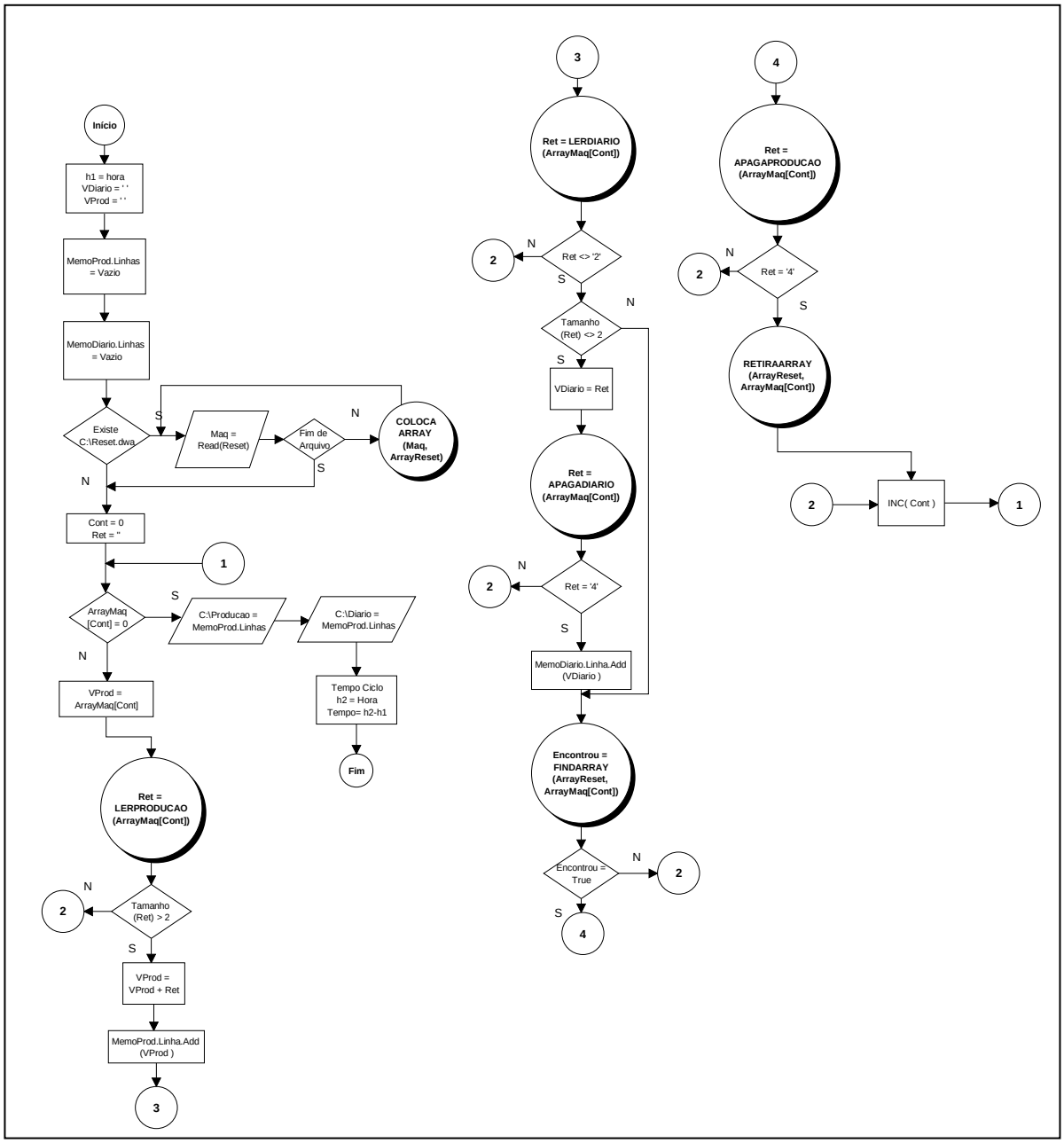
processo, fica na espera do pressionamento de algum botão disponível. Abaixo, na figura 6.4 encontra-se a especificação do processo principal.

Figura 6.4 – Processo Principal



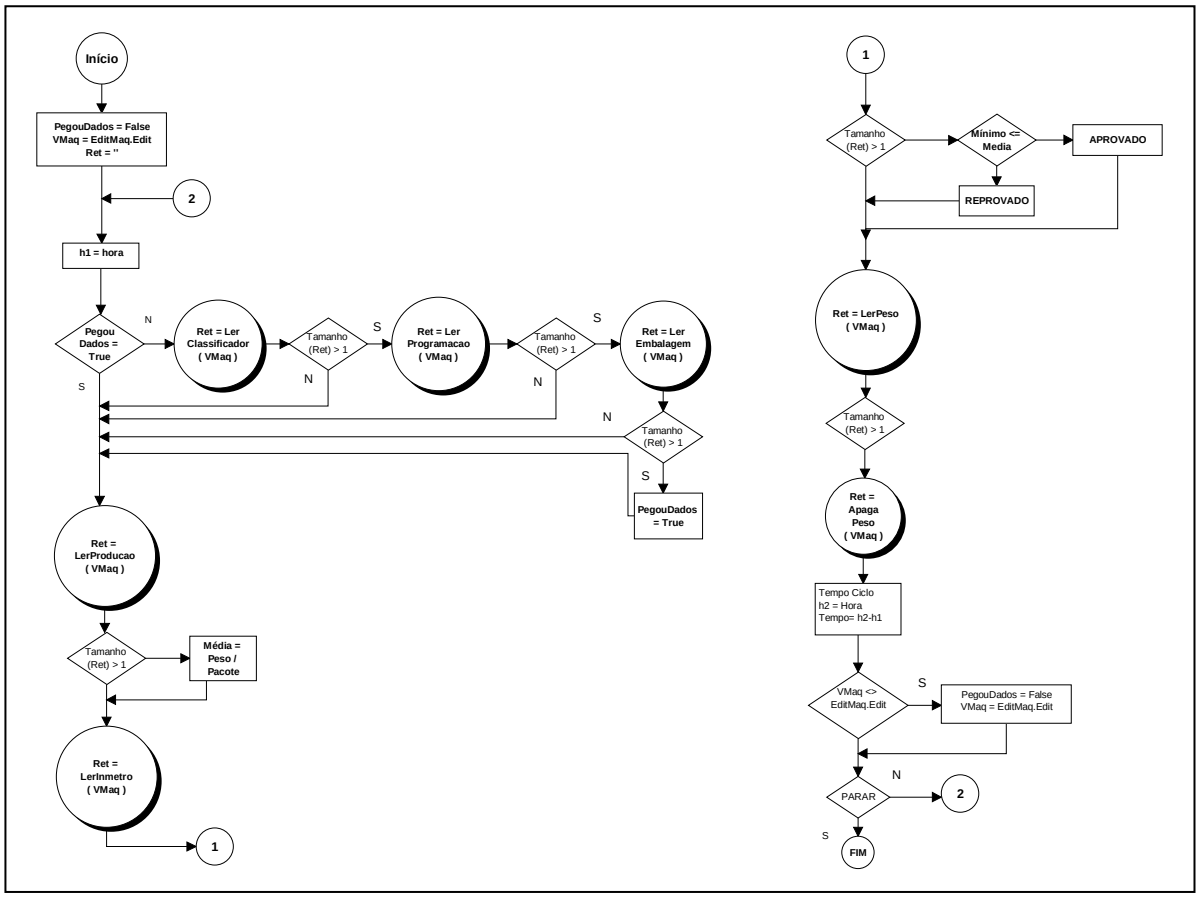
No processo de produção (figura 6.5), primeiramente inicializa-se um *array* de máquinas que devem ser resetadas pelo protótipo. Após é feita a leitura da produção e leitura de um diário da máquina. É verificado se a máquina em questão está no *array* de *reset*. Se a mesma for encontrada, executa-se o comando de *reset.*

Figura 6.5 – Processo de Produção



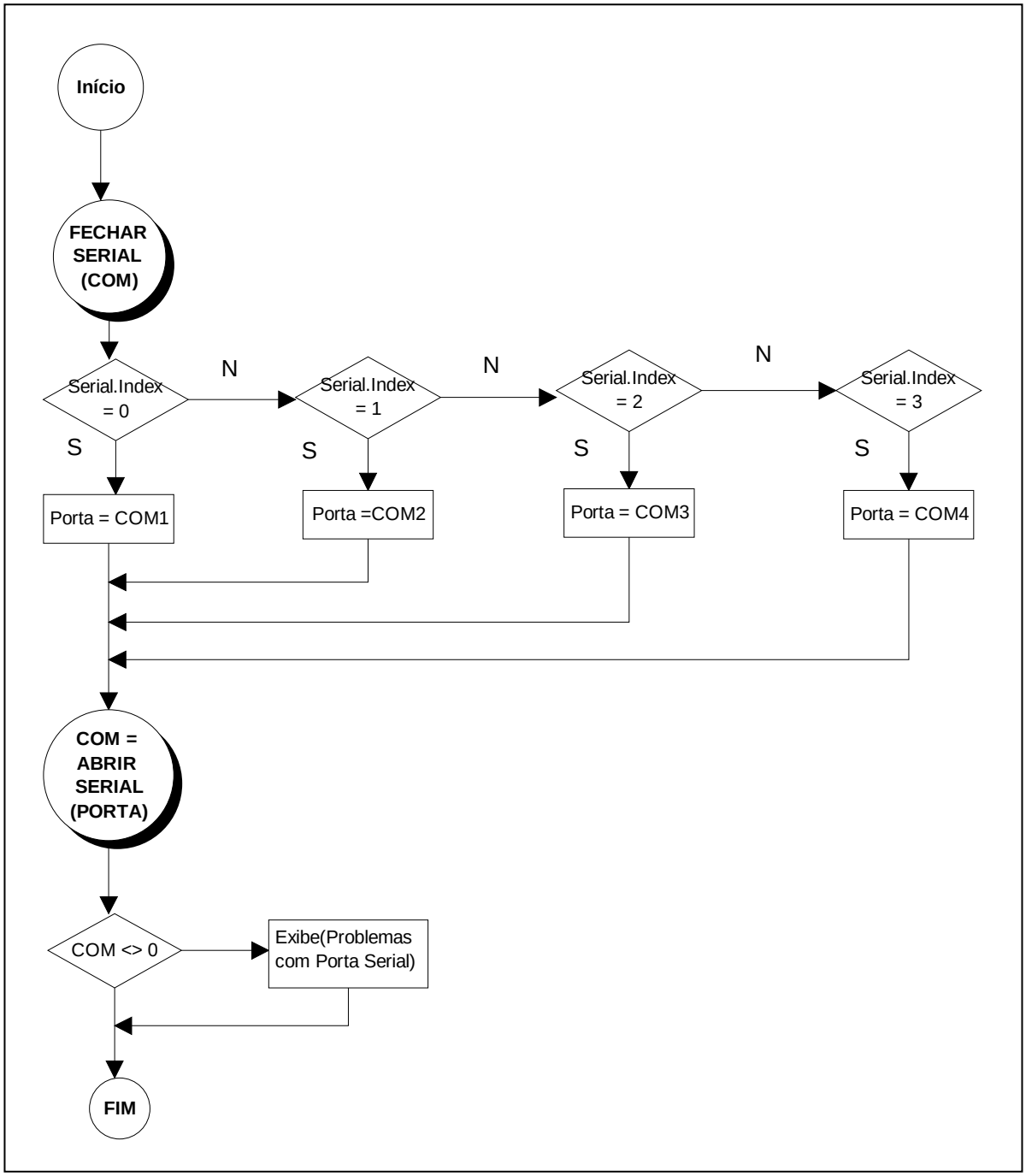
No processo de monitoração (figura 6.6), o protótipo primeiramente lê os dados de programação da máquina. Após é executada uma rotina onde lê-se constantemente os dados de produção, do INMETRO e do último peso da máquina. Nesse processo tem-se ainda as funções de alteração dos parâmetros de programação da máquina e *reset*. No processo de alteração dos parâmetros da máquina, o protótipo irá enviar os dados de classificação e programação que constam na tela do protótipo aos controladores, com os comandos descritos à essa função. O processo de *reset* da máquina também pode ser acionado, como pode ser vista na figura 6.6.

Figura 6.6 – Processo de Monitoração



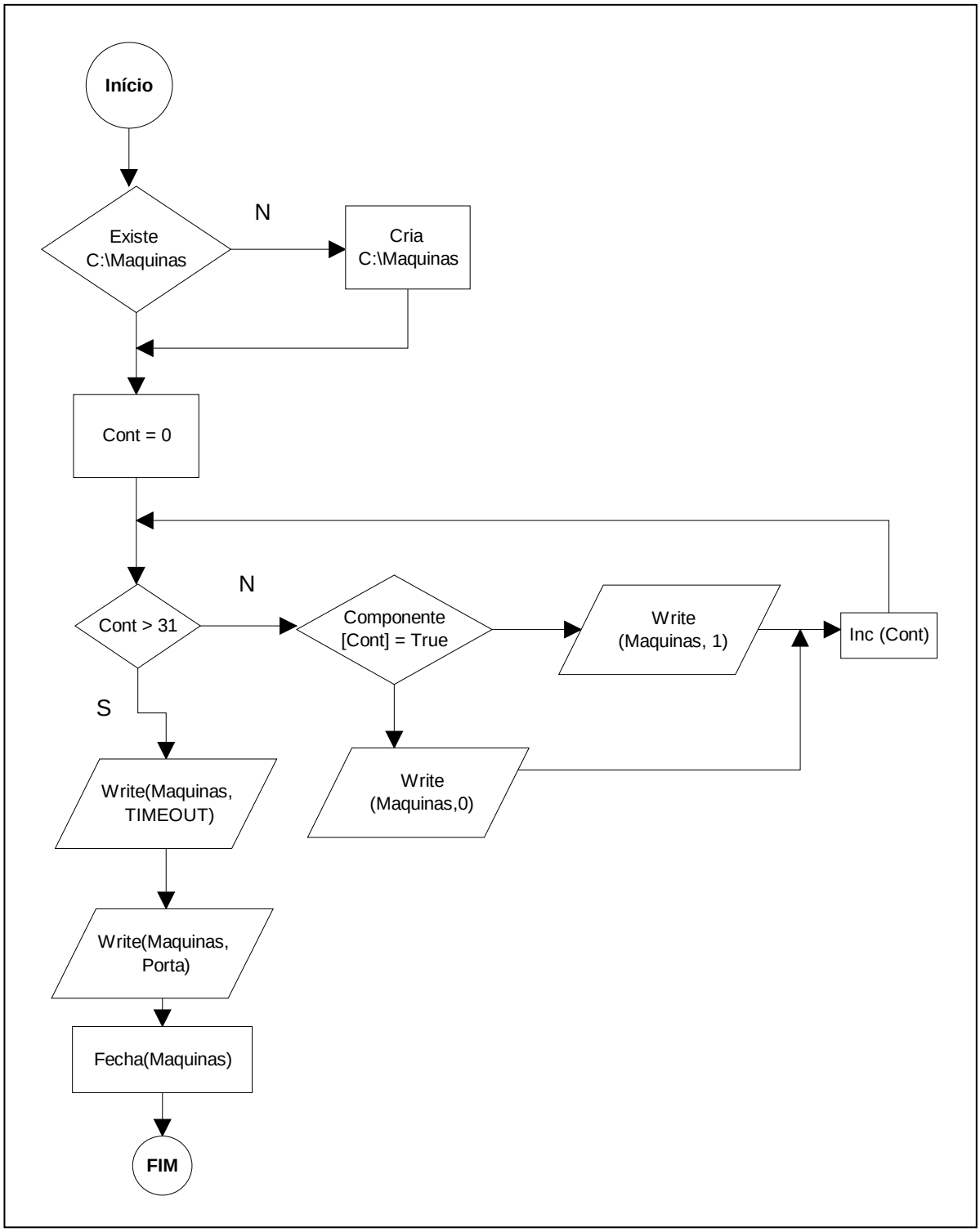
No processo de configuração serial do PC, como demonstra a figura 6.7, o protótipo irá fechar a porta serial atual, e abrir a mesma porta ou uma nova porta, com as configurações escolhidas para *timeout.*

Figura 6.7 – Programação Serial



Para salvar as alterações, o protótipo faz a chamada da função “SalvarSerial”, especificada na figura 6.8.

Figura 6.8 – Salvar Serial



O processo de FecharProtótipo, encerra o protótipo fechando antes a porta serial aberta.

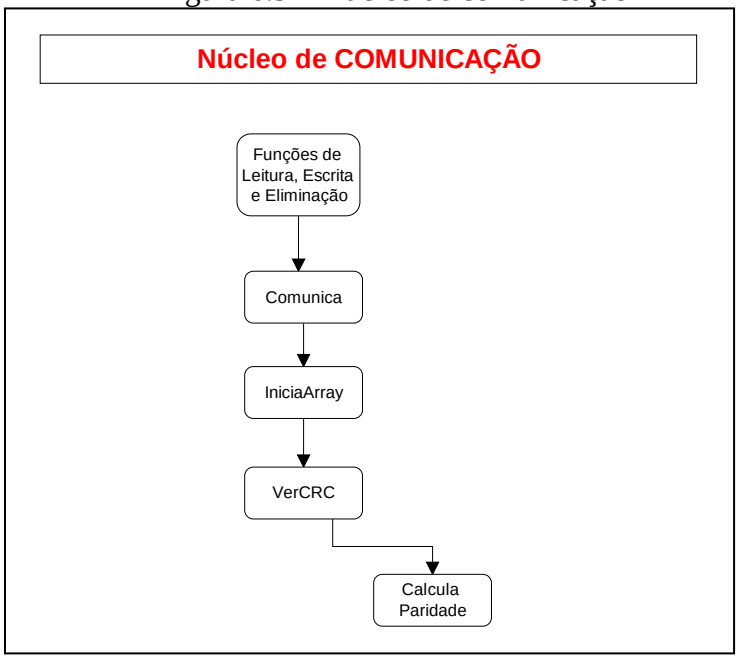
Uma especificação detalhada das subfunções encontra-se no anexo 1 do trabalho.

6.1.4 NÚCLEO DE COMUNICAÇÃO

O núcleo de comunicação, apresentado abaixo (figura 6.9), é chamado por todas as funções de leitura, escrita e eliminação de dados do sistema, ou seja, toda e qualquer comunicação entre protótipo e controlador de peso passa pelo núcleo de comunicação.

Seguindo o fluxo das chamadas de funções a nível de comunicação, tem-se como função principal as funções de leitura, escrita e eliminação de dados. Essas acionam a função de comunicação que é responsável pela inicialização dos *arrays* de entrada e saída de caracteres via interface serial. Após iniciado os *arrays* de entrada e saída é feita a comunicação efetivamente. Recebido o retorno do controlador , o protótipo verifica a paridade da mensagem. A função de verificação de paridade, faz o cálculo da paridade e retorna para a função que à chamou, fechando o ciclo da comunicação entre protótipo e controlador de peso.

Figura 6.9 – Núcleo de comunicação



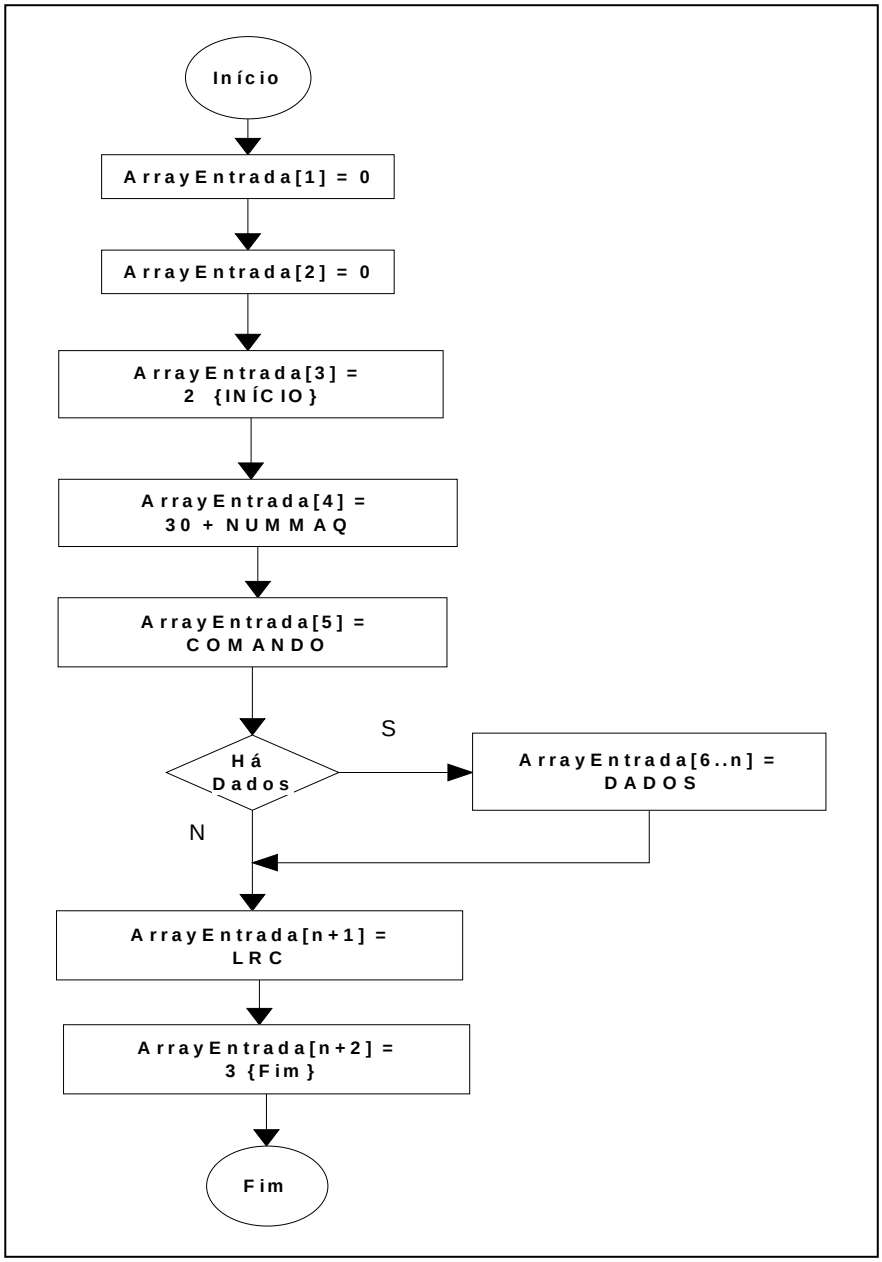
A especificação do núcleo de comunicação encontra-se no anexo 1.

6.1.5 ESPECIFICAÇÃO *MODBUS*

A especificação do protocolo *Modbus* segue o seguinte formato (figura 6.10): Sincronismo + Sincronismo + Byte de inicialização + Número da máquina + Comando + Dados + LRC + Byte de finalização.

Como alterações do *Modbus* original no protótipo, tem-se que o byte de inicialização será o 2, e o byte de finalização será o 3. O números dos controladores serão de 31 a 62. Outra diferença, é que para eliminação de ruídos, envia-se antes da mensagem dois bytes zeros.

Figura 6.10 – Especificação *Modbus* no protótipo



6.1.6 FUNÇÕES DESENVOLVIDAS

Abaixo são descritas todas as funções de leitura, escrita e exclusão de dados aceitas pelo controlador de peso, bem como os parâmetros a serem enviados aos controladores e os retornos que a função recebe.

6.1.6.1 LERPRODUCAO

Essa função lê o acumulador de quilos e pacotes pesados pela máquina. Bem como os pacotes rejeitados com o peso fora do padrão.

Sintaxe.....: LerProducao(NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina para a qual deseja-se pegar os dados de produção

Retorno.....: Essa função retorna uma string do seguinte formato:

(Quilos, Pacotes, RejMais, RejMenos) - QQQQQQPPPPPP++++----

Número de quilos e pacotes pesados e quantidades rejeitadas.

6.1.6.2 LERPESO

Essa função lê o último peso constante na pilha da máquina. O mesmo, após ser lido, deverá ser apagado pela função ApagarPeso, senão permanecerá na pilha e sempre será retornado pela função LerPeso.

Sintaxe.....: LerPeso (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina.

Retorno.....: Essa função retorna uma string do seguinte formato:

PPPP - Último peso registrado

6.1.6.3 LERCLASSIFICADOR

Essa função lê uma string contendo os dados do classificador da máquina. Esses parâmetros podem ser alterados pela função EscreverClassificador

Sintaxe.....: LerClassificador(NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: Essa função retorna uma string do seguinte formato: AAAAaaaaSS

AAAA - Retirar pacote acima de AAAA gramas

aaaa - Retirar pacote após aaaa gramas

SS - Retirar pacote após SS segundos, ou SS posições

6.1.6.4 LERPROGRAMACAO

Essa função lê uma string contendo os dados de programação da máquina. Esses parâmetros podem ser alterados pela função EscreverPrograma ção

Sintaxe.....: LerProgramacao(NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: Essa função retorna uma string do seguinte formato: PPPPppppCC

PPPP - Peso máximo para correção em PPPP gramas

pppp - Peso mínimo para correção em pppp gramas

CC - Fator de correção.

6.1.6.5 LERDIARIO

Essa função lê o último diário constante na fila da máquina. Após a leitura deve-se excluir o diário com a função ApagarDiario.

Sintaxe.....: LerDiario(NumeroMaquina: Integer): string

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: Essa função retorna uma string do seguinte formato:

(Diário, Hora, Minuto, Dia, Mes) - NNN,HH,MM,DD,MM

6.1.6.6 LEREMBALAGEM

Essa função lê o peso bruto que fica armazenado na máquina de empacotar. Pode ser alterado pelo comento EscreveEmbalagem

Sintaxe.....: LerEmbalagem (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: Essa função retorna uma string do seguinte formato:

PPPP - Peso bruto registrado na DWA Check 5P

6.1.6.7 LERINMETRO

Essa função lê o cálculo do INMETRO efetuado pelo controlador de peso com os últimos 32 pacotes pesados.

Sintaxe.....: LerInmetro (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: Essa função retorna uma string do seguinte formato: (MMMMmmmmdd)

MMMM – Peso médio dos últimos 32 pacotes

mmmm – Mínimo segundo a lei do INMETRO

dd – Desvio Padrão

6.1.6.8 APAGARPRODUCAO

Essa função apaga as quantidades acumuladas da Check 5P: quantidade de quilos, quantidade de pacotes, quantidades de pacotes rejeitados acima e abaixo do peso.

Sintaxe.....: ApagaProducao (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: ver tópico 6.1.7

6.1.6.9 APAGARPESO

Essa função apaga um peso da pilha.

Sintaxe.....: ApagaPeso (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: ver tópico 6.1.7

6.1.6.10 APAGARDIARIO

Essa função apaga um diário da pilha.

Sintaxe.....: ApagaDiário (NumeroMaquina: Integer): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

Retorno.....: ver tópico 6.1.7

6.1.6.11 ESCREVERPROGRAMACAO

Essa função envia para a DWA Check 5P uma string contendo os dados para configurar a programação da máquina.

Sintaxe.....: EscreverProgramacao(PortaSerial: String , NumeroMaquina: Integer, ParProgramacao: String): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

ParProgramacao: Conterá todos os valores para modificar os parâmetros do classificador da máquina.

Formato do parâmetro ParProgramacao: PPPPppppCC

PPPP - Peso máximo para correção em PPPP gramas

pppp - Peso mínimo para correção em pppp gramas

CC - Fator de correção.

Retorno.....: ver tópico 6.1.7

6.1.6.12 ESCREVEREMBALAGEM

Essa função envia para a DWA Check 5P uma string contendo o peso bruto do pacote.

Sintaxe.....: EscreverPesoEmbalagem(NumeroMaquina: Integer, PesoBruto: String): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

PesoBruto: Peso do pacote com a embalagem

Formato do parâmetro PesoBruto: NNNN

Retorno.....: ver tópico 6.1.6

6.1.6.13 ESCREVERCLASSIFICACAO

Essa função envia para a DWA Check 5P uma string contendo os dados para programar o classificador da máquina..

Sintaxe.....: EscreverParametrosClassificador(NumeroMaquina: Integer, ParClassificador: String): String

Parâmetros: NumeroMaquina: Parâmetro que conterá o número da máquina

ParClassificador: Conterá todos os valores para modificar os parâmetros do classificador da máquina

Formato do parâmetro ParClassificador: AAAAaaaaSS

AAAA - Retirar pacote acima de AAAA gramas e abaixo de aaaa gramas.

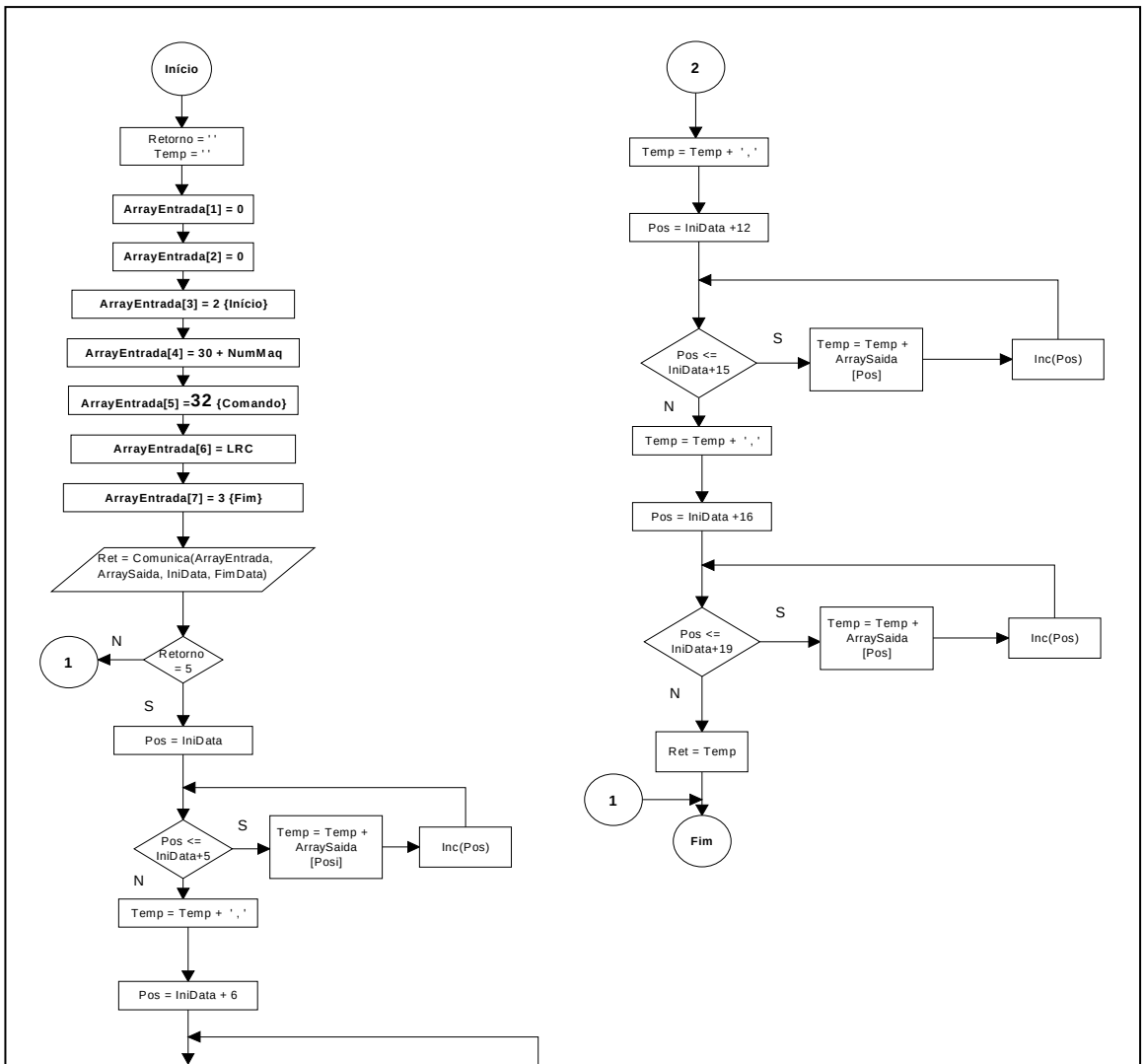
SS - Retirar pacote após 0.SS segundos, ou SS pacotes pesados.

Retorno.....: ver tópico 6.1.7

Abaixo será especificada uma função de leitura, uma função de deleção de dados e uma de escrita nos controladores. As escolhidas foram as funções: LerProducao, ApagaProducao e EscreveClassificador. Todas as demais funções seguem o mesmo modo de especificação, variando apenas nos dados enviados ou recebidos, e no campo comando. Não esquecendo o cálculo de LRC que será diferente para cada mensagem. A especificação das demais funções encontra-se no anexo 1.

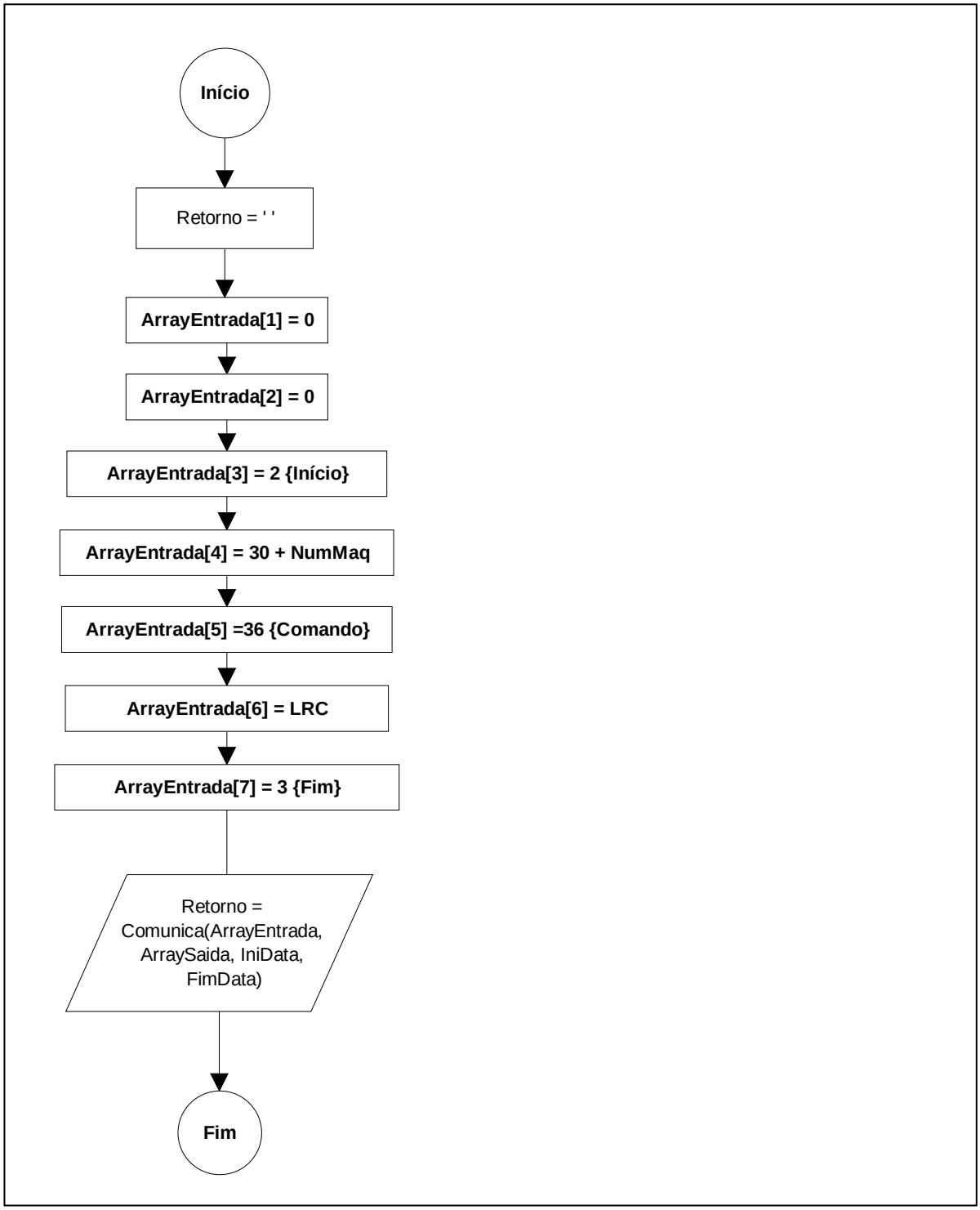
Na figura 6.11, tem-se a especificação da função de leitura que irá efetuar a leitura da produção nos controladores de peso.

Figura 6.11 – Função de Leitura



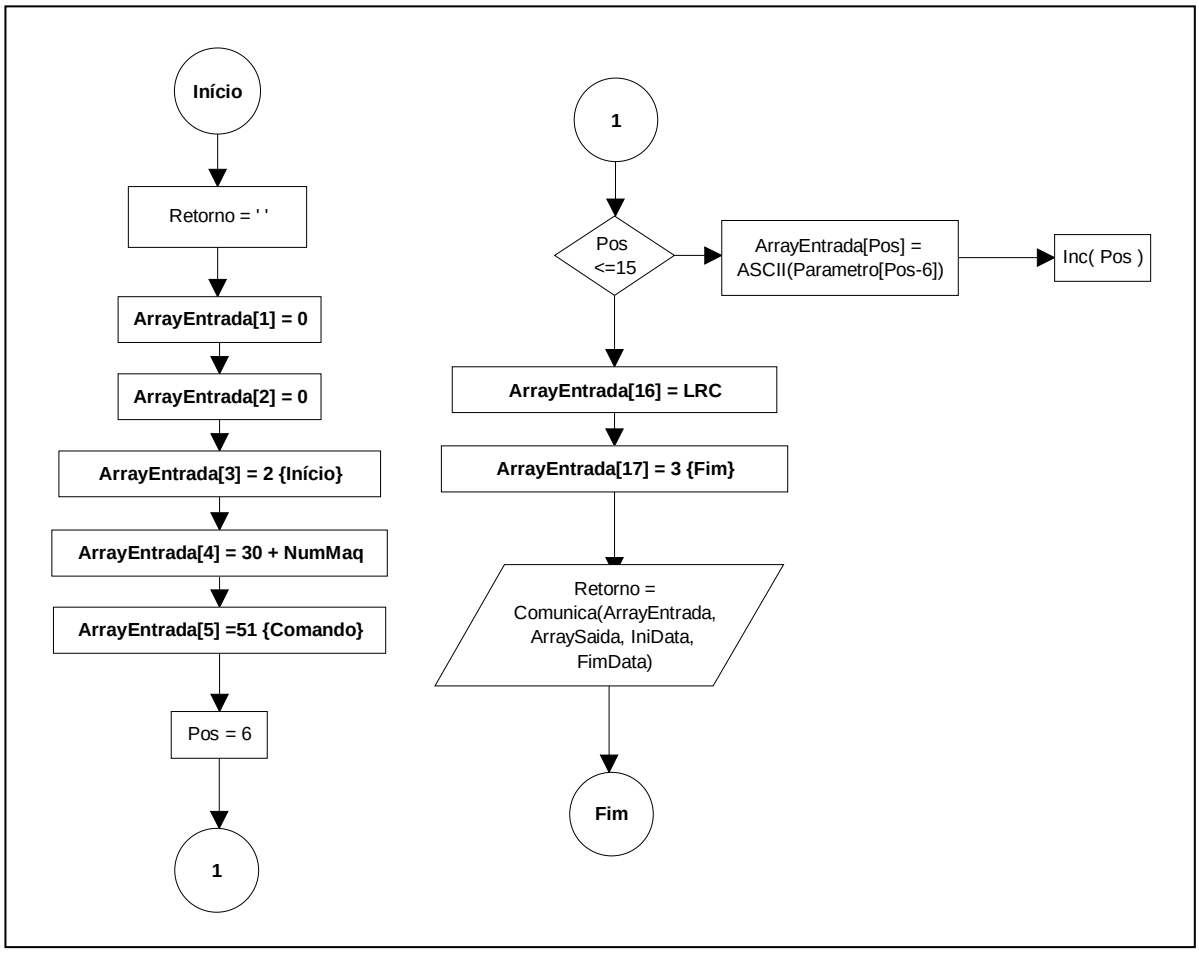
Na figura 6.12, tem-se a especificação da função de exclusão de dados que irá efetuar o zeramento dos dados de produção nos controladores de peso.

Figura 6.12 – Função de exclusão de dados



Na figura 6.13, tem-se a especificação da função de escrita de dados que irá efetuar a escrita dos dados de classificação nos controladores de peso.

Figura 6.13 – Função de escrita de dados



6.1.7 RETORNOS DAS FUNÇÕES

Os retornos das funções desenvolvidas podem ser os seguintes:

a) 0 (zero) - houve problema na porta serial;
b) 1 (um) – erro de *timeout*;
c) 2 (dois) – sem novos dados;
d) 3 (três) – erro de recepção da Check 5P;
e) 4 (quatro) – último comando executado com sucesso;
f) 5 (cinco) – com novos dados;
g) 6 (seis) – erro de paridade;
h) 7 (sete) – erro de continuidade
i) 8 (oito) – erro de dados

Abaixo são descritos os retornos e o motivo pelo qual os mesmos ocorrem.

6.1.7.1 PROBLEMAS NA PORTA SERIAL - RETORNO “0”

Esse retorno é geralmente causado quando se está tentando ler ou enviar mensagens por uma porta serial não conectada ao conversor de rede 485. Outro motivo, é o fato de haver conflito com a porta serial selecionada. Outro provável motivo do retorno zero, pode ser gerado por defeitos físicos na porta (porta queimada). Um ponto importante também é verificar se a porta está instalada no *Windows*.

6.1.7.2 ERRO DE TIME OUT - RETORNO “1”

Os erros de *timeouts* são bem característicos em uma comunicação serial. Trata-se de um tempo pelo qual o sistema fica em estado de espera para receber dados, porém os dados não são enviados durante esse intervalo. Na rede DWA, a prioridade do equipamento é o controle do peso, portanto, há intervalos de aproximadamente 10ms, no qual a máquina não atende a rede. Então é aceitável que algumas vezes esses erros ocorram. Para tanto, basta

efetuar a leitura novamente. Do lado do PC, após o envio da mensagem, espera-se por um tempo de aproximadamente 300ms para retorno da resposta. Se a mesma não vier nesse intervalo, ocorre um erro de *timeout*.

Se houver problemas de *timeout* constante, as causas mais prováveis são:

a) o controlador estar desligado;
b) há problemas com o CI (circuito integrado) do controlador que processa as mensagens recebidas via rede;
c) rompimento do cabeamento, ou curto no mesmo.

6.1.7.3 SEM NOVOS DADOS – RETORNO “2”

Lembra-se que o controlador de peso possui um buffer de 32 posições para armazenamento dos pesos coletados, e um buffer de 32 posições para o armazenamento dos diários inseridos via teclado do controlador. Se esses buffers estiverem vazios, e tenta-se ler um peso ou um diário, o sistema retorna a string “2”, o que significará que o estrutura está vazia, sem nenhum dado registrado.

6.1.7.4 ERRO DE RECEPÇÃO CHECK5P - RETORNO “3”

Esse erro ocorre quando a Check 5P recebe um comando e não consegue interpretá-lo. Isso é possível quando ocorre algum ruído na linha que danifique as informação. Tal erro é detectado pelo cálculo de LRC do controlador de peso.

6.1.7.5 ÚLTIMO COMANDO EXECUTADO – RETORNO “4”

Esse retorno se obtém somente quando se efetua os comandos de exclusão ou escrita do controlador de peso DWA Check 5P. O Código 4 (quatro), significa que o comando foi compreendido pela máquina e executado com sucesso

6.1.7.6 COM NOVOS DADOS – RETORNO “5”

Esse é um retorno interno da biblioteca. Dificilmente será apresentado ao usuário. Porém é usado para o cálculo do desempenho da rede. As funções internas do sistema utilizam esse retorno para verificar as mensagens do controlador.

6.1.7.7 ERRO DE PARIDADE – RETORNO “6”

Mensagem que ocorre quando o protótipo detecta um erro de paridade na chegada das informações ou quando o controlador retorna uma mensagem específica identificando o erro na mensagem de envio.

6.1.7.8 ERRO DE CONTINUIDADE – RETORNO “7”

Toda mensagem possui um caracter de início e outro de finalização da mensagem. Quando após a leitura da porta serial, esses caracteres não são encontrados pelo protótipo, o sistema identifica como um erro de continuidade dos dados.

Leva-se em consideração que o buffer de leitura e escrita na porta serial são de 128 bytes, sendo que se houver muito ruído na linha, os caracteres de início e fim da mensagem podem não estar no buffer.

6.2 IMPLEMENTAÇÃO

Nesse ponto serão mostradas algumas técnicas e ferramentas utilizadas para a implementação do protótipo, bem como a sua operacionalidade e forma de funcionamento. Algumas etapas da implementação consideradas importantes nos processos apresentados também serão citadas.

A implementação completa do protótipo encontra-se no anexo 2.

6.2.1 AMBIENTE DE IMPLEMENTAÇÃO

A implementação do protótipo será no ambiente Delphi. O acesso a porta serial do PC será via API do Windows para comunicação com o equipamento CHECK 5P da DWA, utilizando os padrões RS232C e RS485 como meios de transmissão. Foi escolhido ambiente Delphi, devido a sua facilidade de interação com as funções de API do Windows, e também pela facilidade de tratamento de ponteiros, que são necessários para a utilização de certas funções API, como por exemplo a abertura da porta serial, transmissão e recepção de dados pela mesma, entre outras.

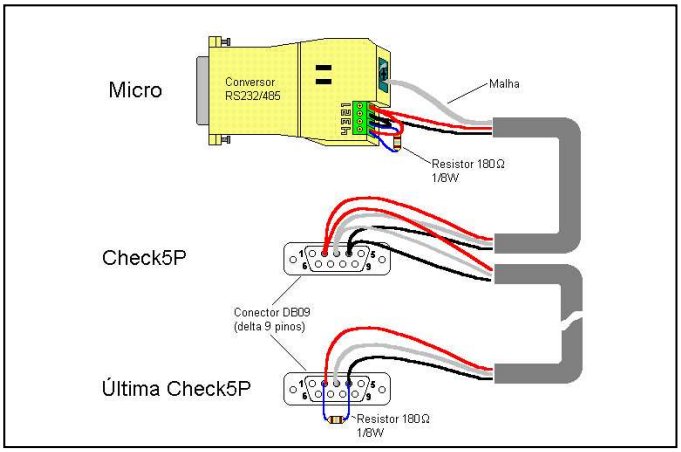
6.2.2 RECURSOS DE HARDWARE NECESSÁRIO

Para funcionamento do protótipo se fazem necessários alguns dispositivos e equipamentos que compõem a rede industrial aqui descrita, ou seja, são necessários um PC rodando o protótipo com RS232C, um conversor de RS232C-RS485 e vice-versa, quatro

metros de cabo Furukawa Fisdata 2 x 24 AWG blindado para conexão entre PC, controlador de peso e um simulador de máquina de empacotar.

Em relação ao cabeamento, segue abaixo, na figura 6.14, a especificação da pinagem e conexão necessária para que a instalação física torne possível o envio e recebimento das mensagens.

Figura 6.14 – Pinagem da Rede



6.2.3 PROTÓTIPO

Basicamente, a função do protótipo é enviar funções via rede para os controladores de peso, receber a resposta dos mesmos e guardar essas informações em um arquivo texto para que programas de terceiros abram esses arquivos e possam pegar tanto os dados de produção (arquivo producao.dwa), acumulados no controlador de peso, como os diários (arquivo diario.dwa) que foram armazenados via teclado do controlador.

Outra função, é ler um arquivo de reset (arquivo reset.dwa) para apagar os dados de programação dos controladores. O arquivo conterá em cada linha um número, o qual será armazenado em um *array*. Quando for efetuado o comando para leitura da rede, o sistema também verifica se a máquina encontra-se no *array* de *reset.* Estando lá, o sistema primeiramente lê os dados, depois apaga-os do controlador, e se o comando obteve sucesso, retira o número da máquina do *array*.

Outra função é um monitoramento *on-line* dos pacotes, quantidades e processos que está ocorrendo em cada máquina. Nesse monitoramento, o sistema busca a aproximadamente cada 400 milisegundos, informações como, quantidade de pacotes e quilos produzidos, quantidades rejeitadas, a programação do controlador de peso, o cálculo do INMETRO, se os pacotes atuais estão aprovados segundo o critério do INMETRO ou não, e também o peso do último pacote que o controlador processou.

Um detalhe que deve ser observado, é que como trata-se de um protótipo, os processos são iniciados com um clique do mouse sobre os botões. Porém, o sistema deverá ser programado com um *timer* de 1 (um) minuto para a coleta das informações, isso quer dizer, a cada minuto, o sistema inicia automaticamente o processo de leitura da rede e armazenamento em arquivo texto da produção e diário.

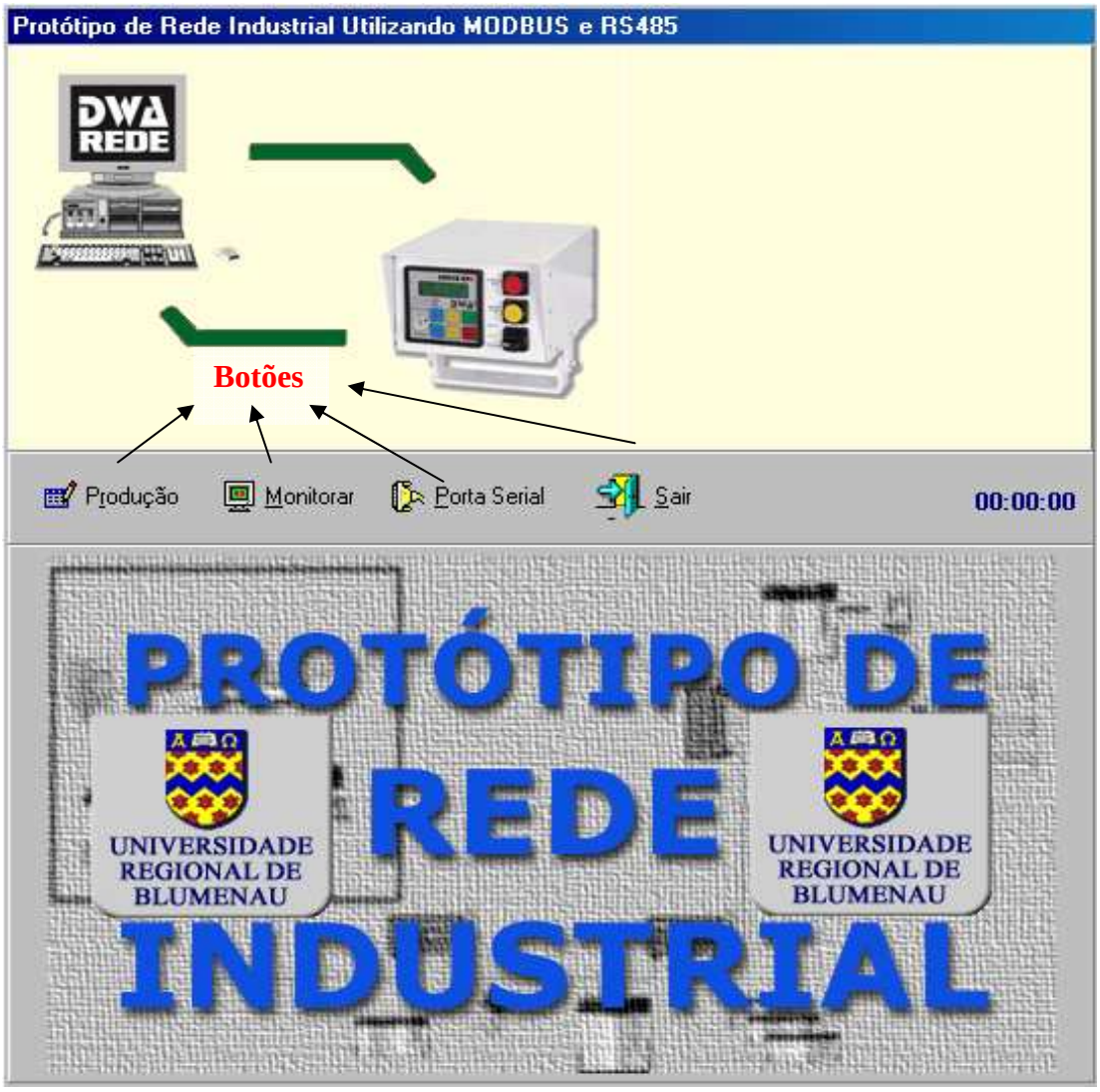
Outro ponto relevante, é que ao iniciar o processo de monitoramento individual de um equipamento, automaticamente para-se o *timer* de um minuto (responsável pela coleta dos dados de produção), fazendo com que o sistema fique exclusivamente executando a tarefa de monitoramento, e não mais armazenamento dos dados de produção. Ao sair da tela de monitoramento o timer de 1 um minuto deverá ser reiniciado.

Segue abaixo, a apresentação das telas do protótipo.

6.2.3.1 TELA PRINCIPAL DO SISTEMA

Na figura 6.15 é apresentada a tela de entrada do protótipo. Nessa tela vê-se os botões de produção, monitorar, porta serial e sair.

Figura 6.15 – Tela de entrada do protótipo



Como já visto as funções principais do sistema, basta “clicar” em um dos botões apresentados na figura 6.15 para se ter acesso a busca de informações. Através do botão “produção”, tem-se acesso a busca dos dados de produção e diários, monitoramento da

máquina, através do botão “monitorar” e através do botão “porta serial” tem-se acesso a alteração dos parâmetros de comunicação.

A nível de implementação, o quadro abaixo (Quadro 6.1) demonstra algumas partes da inicialização do sistema. A função descrita abaixo, é inicializada quando sistema é executado retornando a configuração serial previamente gravada em arquivo. A função completa encontra-se nos anexo 2.

Quadro 6.1 – Retorno da configuração da porta serial

```
procedure TFormColeta.FormShow(Sender: TObject);
. . .
    AssignFile(VMaquina,'C:\Maquinas.Ini') {Lê arquivo de config};
    Try
       Reset(Vmaquina);
       . . .
      {Retorna os TimeOuts gravados}
       Try
          Readln(VMaquina,Rascunho);
          TO_LeituraTotal:= StrToInt(Rascunho);
          Readln(VMaquina,Rascunho);
          TO_LeituraByte:= StrToInt(Rascunho);
            Readln(VMaquina,Rascunho);
          TO_LeituraAcrescimo:= StrToInt(Rascunho);
          Readln(VMaquina,Rascunho);
          TO_EscritaTotal:= StrToInt(Rascunho);
          Readln(VMaquina,Rascunho);
          TO_EscritaByte:= StrToInt(Rascunho);
          Readln(VMaquina,Rascunho);
          TimeSleep:= StrToInt(Rascunho);
       Except
          TO_LeituraTotal:= 300;
          TO_LeituraByte:= 50;
          TO_LeituraAcrescimo:= 50;
          TO_EscritaTotal:= 3;
          TO_EscritaByte:= 3;
          TimeSleep:= 1;
       End;
        {Retorna a Porta Serial}
       Readln(VMaquina,Rascunho);
       {Coloca a porta na serial escolhida}
       RadioGroupPortaSerial.Itemindex:=StrtoInt(Rascunho);
       Case RadioGroupPortaSerial.ItemIndex of
          0: Porta:= 'COM1';
          1: Porta:= 'COM2';
          2: Porta:= 'COM3';
          3: Porta:= 'COM4';
```

Outra função inicializada na execução do sistema é a função de abertura da porta serial, descrita em partes no quadro 6.2. A função completa encontra-se descrita nos anexo 2.

Quadro 6.2 – Abertura da porta serial

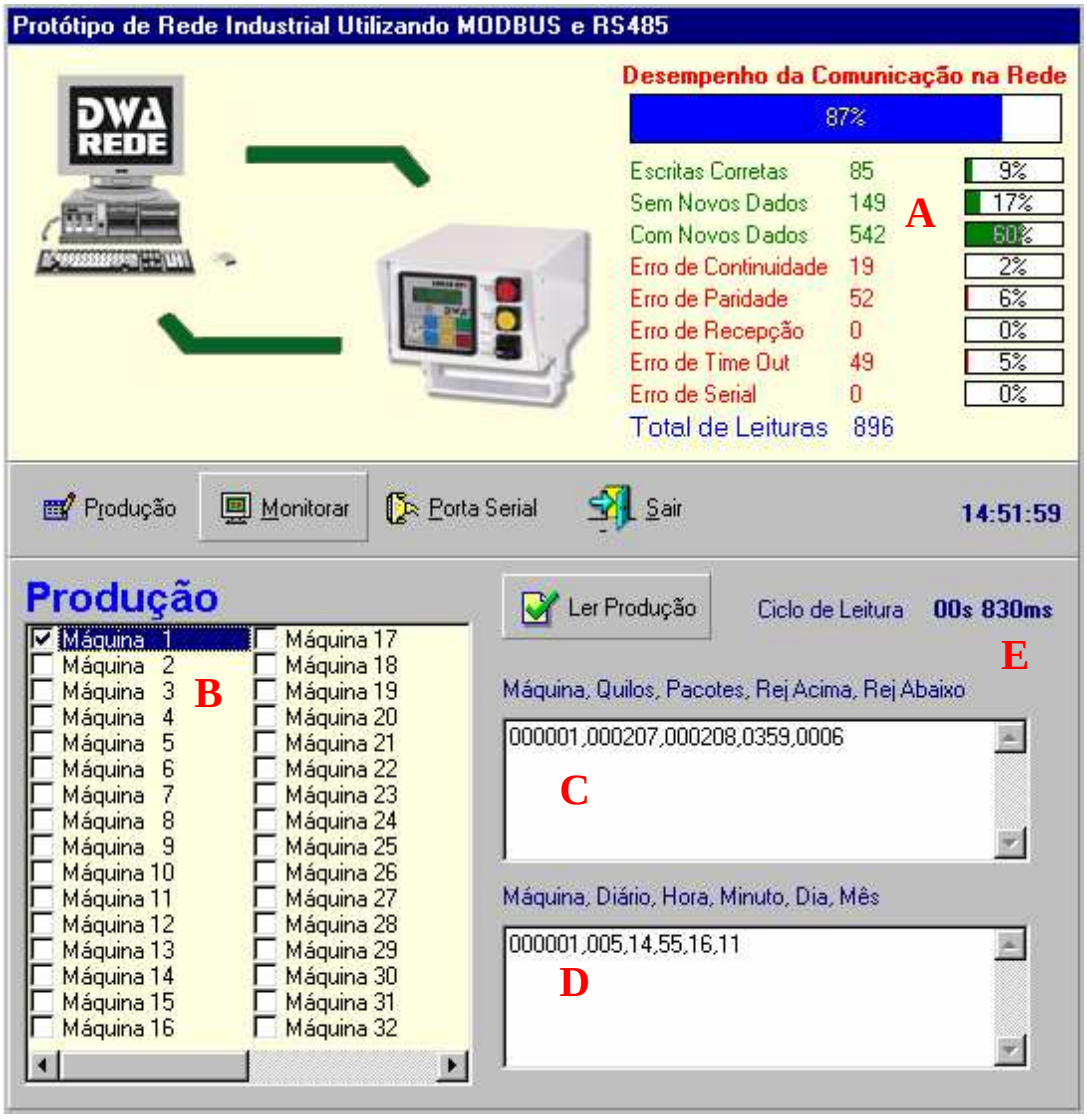
```
Function AbrirSerial(Serial: String): THandle;
. . .
   {Fecha a porta se estiver aberta}
   CloseHandle( hComm );

   {Abre a porta, para leitura e gravação}
   hComm := createfile (  PChar(Serial) ,generic_WRITE or
            generic_READ,1,nil,open_existing,file_flag_overlapped,0);
   If hComm > 0 Then
   Begin
      {Seta as configuracoes da porta}
      setupComm ( hcomm,128,128);
         DCB.BaudRate   :=9600;
         DCB.Parity     :=NOPARITY;
         DCB.ByteSize   :=8;
         DCB.StopBits   :=ONESTOPBIT;
         DCB.XOnChar    :=Char(11);
         DCB.XOffChar   :=Char(13);
         DCB.XOnlim     :=20;
         DCB.XOfflim    :=20;
         DCB.ErrorChar  :=CHAR(03);
```

6.2.3.2 TELA DA PRODUÇÃO DO SISTEMA

Na figura 6.16 é apresentada a tela de produção do protótipo.

Figura 6.16 – Tela de produção



Como pode-se observar na figura 6.16, é apresentada a tela responsável pelas operações de busca de produção e diários no controlador de peso DWA Check 5P. A tela é composta pelos itens abaixo descritos.

No item “A”, é apresentado uma estatística sobre o desempenho da comunicação na rede. Cada mensagem que o PC envia para os controladores de peso, passam necessariamente por uma função de comunicação, apresentado na especificação do protótipo. Nessa função é observado o retorno de cada mensagem e calculado os dados de desempenho da rede. Em suma, a cada mensagem enviada, incrementasse o total de leituras e os demais dados são representados pela sua expressividade sobre o total de leituras efetuadas. A implementação dos dados de desempenho é mostrada no quadro 6.3.

Quadro 6.3 – Cálculo do desempenho da rede

```
Function Comunica(Entrada: TArrayByte; Var Saida: TArrayByte; Var Ini: Integer;
        Var Fim: Integer): String;

. . .
{Alimenta os dados estatísticos}
  Estatistica:= StrToInt(RetCOM);
  Case Estatistica Of
    0: FormColeta.EditSerial.Text       :=
       IntToStr(StrToint(FormColeta.EditSerial.Text) + 1);
    1: FormColeta.EditTimeOut.Text      :=
        IntToStr(StrToint(FormColeta.EditTimeOut.Text) + 1);
    2: FormColeta.EditSemNovosDados.Text :=
       IntToStr(StrToint(FormColeta.EditSemNovosDados.Text) + 1);
    3: FormColeta.EditErroRecepcao.Text  :=
       IntToStr(StrToint(FormColeta.EditErroRecepcao.Text) + 1);
    4: FormColeta.EditComandoCerto.Text  :=
        IntToStr(StrToint(FormColeta.EditComandoCerto.Text) + 1);
    5: FormColeta.EditNovosDados.Text    :=
       IntToStr(StrToint(FormColeta.EditNovosDados.Text) + 1);
    6: FormColeta.EditParidade.Text      :=
       IntToStr(StrToint(FormColeta.EditParidade.Text) + 1);
    7: FormColeta.EditContinuidade.Text  :=
       IntToStr(StrToint(FormColeta.EditContinuidade.Text) + 1);
  End;
  FormColeta.EditTotalLeituras.Text :=
     IntToStr(StrToint(FormColeta.EditTotalLeituras.Text) + 1);
. . .
```

Basicamente o cálculo de cada item dentro do desempenho da rede, é a divisão do seu montante pelo total de leituras, ou seja, se *timeouts* tem 50 retornos, num universo de 100 leituras, o seu percentual de representatividade é de 50% . Para cálculo do desempenho total da rede, soma-se os retornos representados por “total de escritas corretas”, “sem novos dados” e “com novos dados”, e divide-se pelo total de leituras.

No item “B”, são apresentadas as máquinas que estão instaladas em rede as quais o protótipo deverá efetuar a leitura de produção e diários. Ou seja, se houver 7 (sete) máquinas na rede, a sugere-se que as mesmas sejam nomeadas de forma ordenada, começando-se com 1. Com isso tem-se na rede máquinas de 1 a 7. Deixando habilitada as máquinas de 1 a 7 no “grid” representado por “B”, ao ser iniciada a operação de leitura das máquinas, o sistema somente irá pegar os dados das máquinas que estiverem habilitadas.

No item “C”, após cada leitura de produção aparecerá os dados enviados pelo controlador de peso na ordem do item “C”. O mesmo ocorre com o item “D”. O sistema irá ler um diário por vez armazenado na pilha do controlador e apresentá-lo em tela. A implementação da busca dos dados de produção pode ser observada no quadro 6.4 , abaixo.

Quadro 6.4 – Busca e gravação dos dados de produção

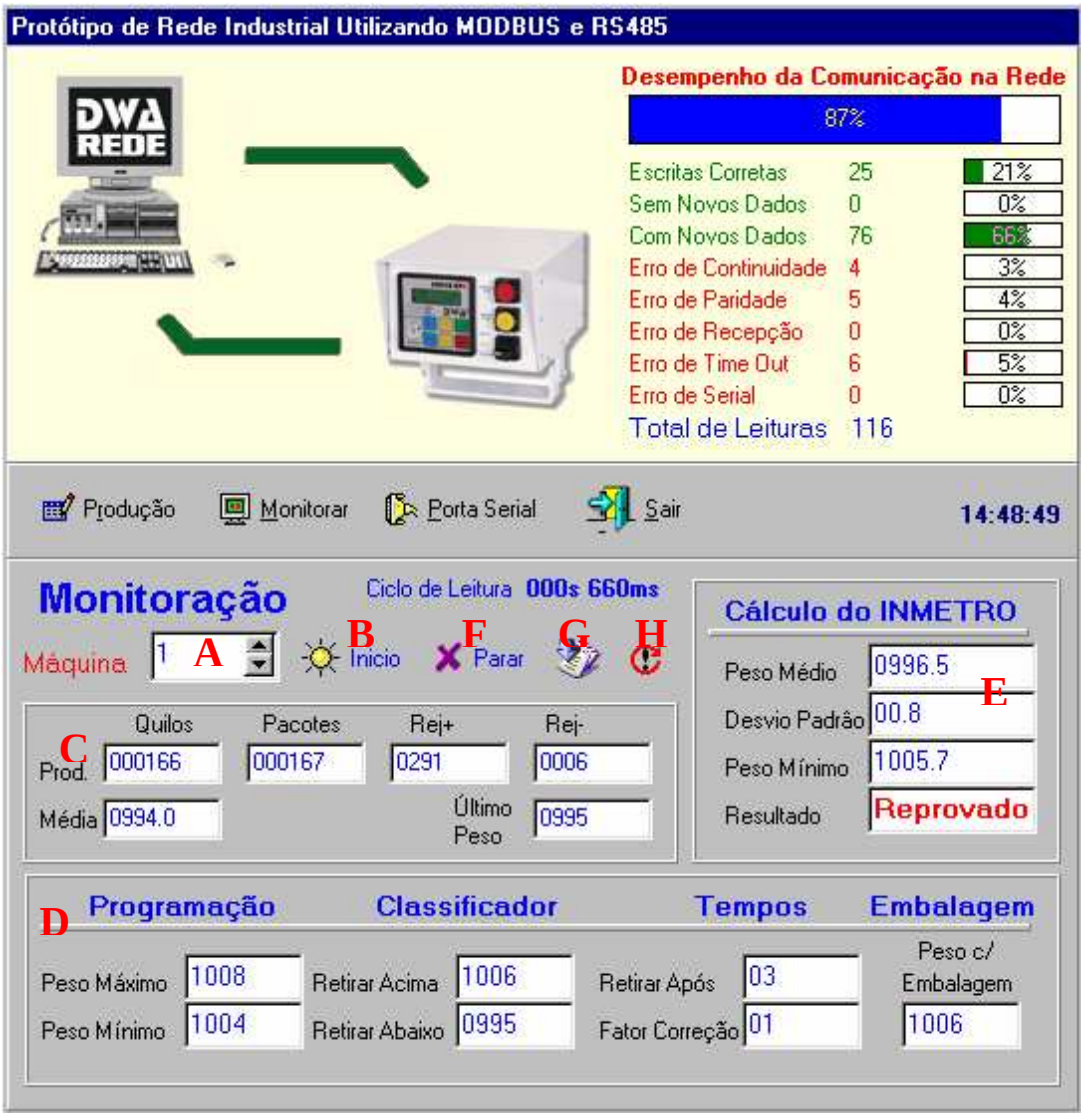
```
procedure TFormColeta.BtLerClick(Sender: TObject);
. . .
{Coloca o Numero da maquina na string}
    Producao:= FormatFloat('000000',MaquinasHabilitadas[Descritor]) + ',';
    {Le os dados de producao, tenta cinco vezes}
    Retorno:= LerProducao(MaquinasHabilitadas[Descritor]);
    {Grava os dados de qtde e quilos no array de producao}
    Producao:= Producao + Retorno;
    {Se a variavel tiver esse tamanho, é por que leu tudo}
    If (Length(Producao) = 30)Then
    Begin
      {Grava no arquivo de producao os dados lidos}
      Memoproducao.Lines.Append(producao);
. . .
```

Após completar-se o ciclo de leitura de todas as máquinas, o protótipo armazena os dados de produção no arquivo “LEITURA.DWA” e os dados de diários em “DIARIO.DWA”. Também exibe em tela o tempo gasto para completar o ciclo, apresentado no item “E”.

6.2.3.3 TELA DE MONITORAÇÃO DO SISTEMA

Na figura 6.17 é apresentada a tela de monitoração individual de cada controlador de peso.

Tela 6.17 – Tela de monitoração



Na monitoração individual, o sistema fará um monitoramento on-line do processamento de peso efetuado no controlador. A aproximadamente 400ms, é atualizada todas as informações apresentadas na figura 6.17.

No item “A”, é onde o usuário escolhe a máquina que deseja monitorar. Após a escolha da máquina, deve-se clicar no botão “INICIO” (item B), para iniciar a coleta de informações, descritas nos itens “C”, “D” e “E”.

No item “F”, apresentado o botão “PARAR”, finaliza o monitoramento da máquina escolhida. Sua operação consiste em setar um flag como verdadeiro, fazendo com que o sistema interrompa o ciclo de leituras.

A qualquer momento, pode-se pressionar os botões representados pelos itens “G”, que é a escrita dos dados de classificação, programação, tempos e embalagens. Para exemplificar, ao iniciar o monitoramento da máquina, o sistema busca a programação atual e apresenta em tela. A programação pode ser alterada e enviada para o controlador. Da mesma forma, o botão representado pelo item “H”, envia para o controlador um comando de reset, ou seja, apaga os dados de produção do controlador.

A nível de implementação, é descrito no quadro 6.5 uma parte da função de monitoração. Especificamente a coleta dos dados de produção e cálculo da média

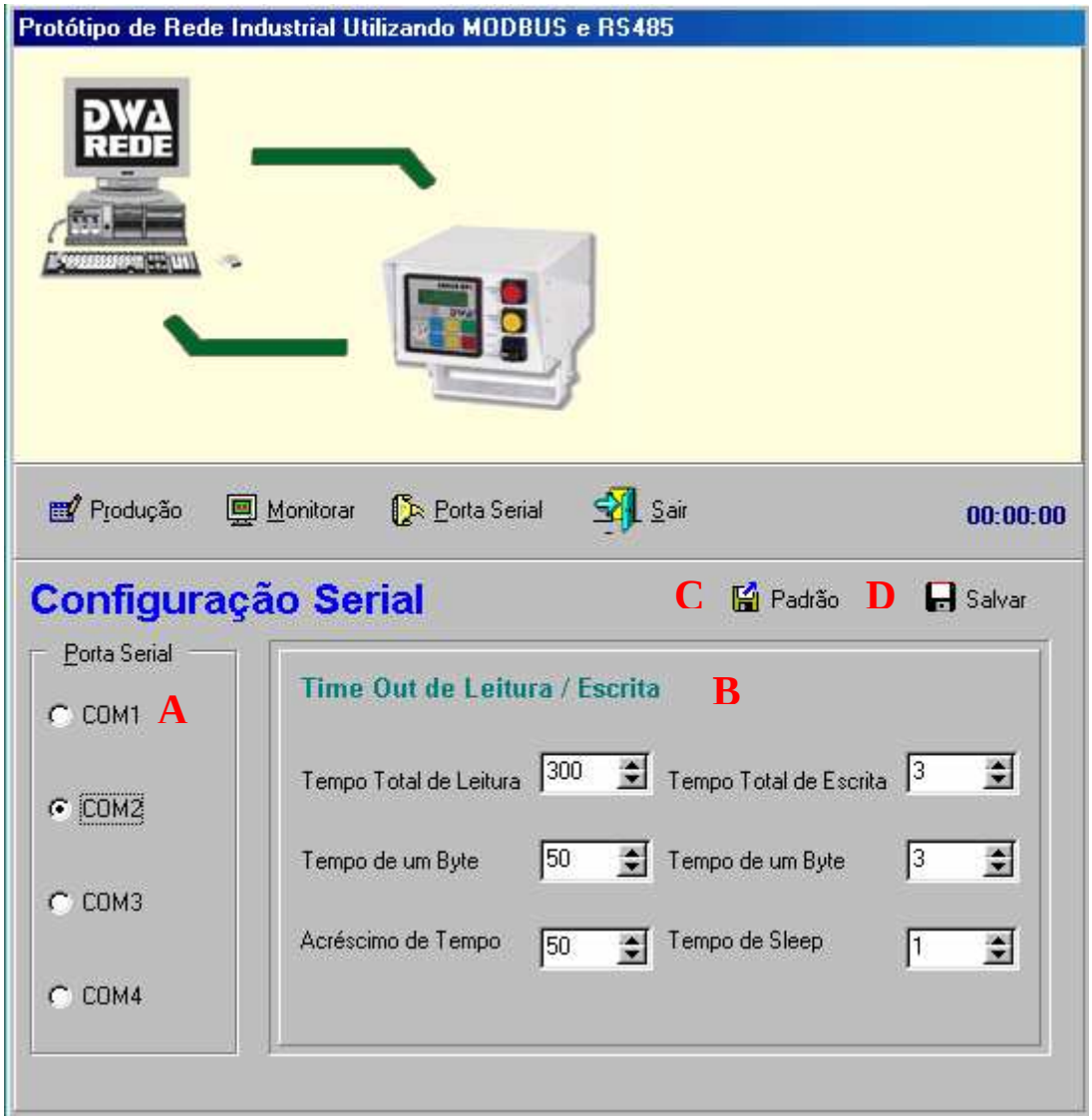
Quadro 6.5 – Função de monitoração

```
procedure TFormColeta.BtInicioClick(Sender: TObject);
. . .
  {Pede os Dados de Producao}
   Rascunho := LerProducao(SpinEditmaquina.Value);
   {Exibe os Dados de Producao}
   If Length(Rascunho) > 1 Then
   Begin
     EditQuilos.Text  := Copy(Rascunho,1,6);
     EditPacotes.Text := Copy(Rascunho,8,6);
     EditRejMa.Text  := Copy(Rascunho,15,4);
     EditRejMe.Text  := Copy(Rascunho,20,6);
     Try
       EditMedia.text:= FormatFloat('0000.0',(StrToFloat(EditQuilos.Text) /
                        StrToFloat(EditPacotes.Text)*1000));
     Except
       EditMedia.text:= '0000.0';
     End;
   End;
. . .
```

6.2.3.4 TELA DE CONFIGURAÇÃO SERIAL DO SISTEMA

Na figura 6.18 é apresentada a tela de configuração serial do sistema, bem como a gravação da configuração para posterior retorno.

Figura 6.18 – Tela de configuração serial



Na tela acima, é apresentado a última janela do sistema, onde escolhe-se a configuração serial que mais se adapta às condições físicas da rede. Entende-se por condições físicas os diversos parâmetros que possam influenciar na comunicação serial. Cita-se por

exemplo, ruídos, fazendo com que o tempo de *Sleep* seja maior, longas distâncias, fazendo com que o tempo de total de leitura seja maior também e outros mais.

Antes de verificar a operacionalidade da tela apresentada, deve-se obter algumas informações sobre *timeout*. Como já visto, o tempo de *timeout*, é um intervalo que o sistema fica ouvindo a porta serial para chegada de informações. O termo *"Tempo Total de Leitura*", define o tempo que o sistema deve esperar para chegada do primeiro byte esperado. O "*Tempo de um byte*" é o tempo que se deve esperar no máximo para chegada de um byte após o outro, ou seja, chegou um byte, espera-se "n" milisegundos para o próximo byte. O "*Tempo de Acréscimo*", é um tempo adicional além do tempo total. O "*Tempo de Escrita*" e "*Tempo de um byte de Escrita*" é relativo a comunicação interna entre o processador e a interface serial do micro. Como ambos encontram-se no próprio hardware do PC, a comunicação é em altíssima velocidade, bastando colocar tempos curtos somente como parâmetros. O "*Tempo de Sleep*", refere-se ao tempo de espera após a escrita no controlador de peso. Como padrão, assume-se 1 ms como tempo de Sleep, para chegada do último byte no controlador.

Para se alterar os itens, basta colocar novos valores nos campos. Como o sistema abre a porta serial com os parâmetros programados ao iniciar o sistema. Deve-se após a alteração, salvar, sair do sistema e entrar novamente. Outra forma, é fazer a troca da porta serial, representado pelo item "A", e voltar a porta correta. Como a troca de porta significa o fechamento da porta atual e abertura de uma nova porta, a mesma será aberta com a configuração atual.

No botão "PADRÃO", assume-se como *timeout* a seguinte configuração.

Tempo Total de Leitura – 300

Tempo de um Byte de Leitura – 50

Tempo de Acréscimo – 50

Tempo Total de Escrita – 3

Tempo de um Byte de Escrita – 3

Tempo de *Sleep* - 1

No quadro 6.6, é apresentado a rotina que padroniza o *timeout* da porta serial.

Quadro 6.6 – Função de padronização de *Timeout*

```
procedure TFormColeta.BtDefaultClick(Sender: TObject);
. . .
{Retorna em tela a configuração padrão}
   SpinLeituraTotal.Value:= 300;
   SpinLeituraByte.Value:= 50;
   SpinLeituraAcrescimo.Value:= 50;
   SpinEscritaTotal.Value:= 3;
   SpinEscritaByte.Value:= 3;
   SpinSleep.Value:= 1;
. . .
```

No botão “SALVAR”, o sistema irá escrever o arquivo “C:\MAQUINAS.DWA”, gravando toda a configuração escolhida.

O botão “SAIR”, encerra a sessão do protótipo, fechando a porta serial, cuja função está implementada no quadro 6.7. Lembrando-se que o RTS é utilizado para colocar o meio em modo de transmissão ou em modo de recepção

Quadro 6.7 – Fechamento da porta serial

```
{Recebe a porta serial para ser fechada e liberada a variavel}
procedure FecharSerial(Var Serial: THandle);
begin
  {Retira a linha DTR ou RTS se estiverem setadas}
  escapeCommFunction ( Serial,clrDTR );
  escapeCommFunction ( Serial,clrRTS );

  {Fecha a porta serial liberando a mesma}
  CloseHandle( Serial );
End;
```

6.2.4 TESTES E VALIDAÇÃO DO PROTÓTIPO

O objetivo de interligar os controladores de peso DWA Check 5P ao computador, fornecendo os dados gravados em memória RAM dos mesmos, foi alcançado.

Como já comentado, ressalta-se que o protótipo não efetua a leitura dos controladores de peso em intervalos de tempo, sem que haja o comando dado pelo usuário. Tal decisão foi tomada para que a seqüência de apresentação do protótipo e defesa desse trabalho seja feita de forma didática. Porém a implementação de um *timer* para busca de informações em ciclos de tempo é fácil de ser implementada.

Os testes para se obter a comunicação inicial com o controlador de peso, tornou-se possível após a especificação e implementação do protocolo *Modbus* tanto em linguagem de alto nível, usada na programação do PC, como em linguagem de baixo nível, cita-se o *assembler*, aplicado à programação do controlador de peso.

De início obteve-se algumas dificuldades em relação aos tempos de acesso ao meio RS485, devido a má configuração do *Timeout* da porta serial do PC. Após a fase de acertos ao protótipo, iniciou-se efetivamente a fase de testes, usando-se dois simuladores de máquina de empacotar em funcionamento, e o controlador de peso. Após 48 horas de testes contínuos obteve-se o seguinte resultado em relação ao desempenho da comunicação na rede:

a) escritas corretas 7.97%
b) sem novos dados 11.84%
c) com novos dados 61.85%
d) erro de continuidade 12.72%
e) erro de paridade 0.16%
f) erro de *timeout* 5.45%
g) erro de recepção 0.01%

Foram registradas aproximadamente 393 perguntas por minuto de tráfego na rede, lembrando que cada string na rede possui um tamanho e 32 bytes. A eficiência da rede resultou em 81.66% de dados válidos.

7 CONCLUSÃO

As redes industriais, são uma excelente escolha para empresas que queiram automatizar processos e informatizar informações. Isso significa um maior controle nos processos, evitando perdas tanto de matéria-prima e perdas de tempo e também uma margem de erro das informações praticamente inexistente, já que os dados gerados na máquina não passam por processamento manual.

As várias tecnologiaa para se interligarem equipamentos industriais a um PC fazem da rede industrial um mecanismo de controle acessível em praticamente todo o mundo. Claro que cada região possui sua particularidade e afinidade à certas tecnologias.

O estudo e implementação do protocolo *Modbus*, mostrou a sua abrangência em redes industriais, permitindo as diversas derivações de acordo com a necessidade da aplicação. Obviamente deve-se manter sua estrutura básica para não perder sua padronização. Sua implementação é simples e sua estrutura para formação das mensagem que trafegam via rede é objetiva e clara.

O meio de transmissão RS485 foi satisfatório, mostrando sua capacidade de transmissão imune a ruídos e sua simplificação de cabeamento. Porém, algo que foi verificado, é o cuidado que se deve ter na instalação do cabeamento RS485. Em se tratando de ambientes industriais, deve-se tomar cuidado em deixá-lo isolado de condutores elétricos, manter em local onde não há tráfego de veículos, como empilhadeiras, por exemplo.

Sobre o controlador de peso e o protótipo, algumas observações são relevantes:

a) O módulo de controle de peso Check 5P não responde à rede sempre que questionado, devido a sua prioridade em controlar o peso;

b) A taxa de transmissão é fixa de 9600 bps;

c) Não há comunicação entre os controladores, o que resulta na parada de uma máquina de empacotar se o módulo de controle ocasionar problemas;

d) O controlador de peso é proprietário utilizando um padrão protocolar normativo (*Modbus*). Isso faz com que o protótipo se torne inválido se

forem mudadas as regras de comunicação do controlador de peso.

7.1 EXTENSÕES

As sugestões para futuros trabalhos na área de redes industriais são:

a) criar um protótipo que processe informações trazidas do coletor de dados;

b) estudar novas propostas e tecnologias de redes industriais;

c) viabilizar um núcleo de coleta de dados de equipamentos que utilizem o protocolo *Modbus*. Generalizando o protótipo de forma que ele se adapte a qualquer rede desse padrão;

d) estudo de tecnologias de redes industriais que utilizem TCP como protocolo e a tecnologia Ethernet para ligação direta com uma rede Ethernet.

e) Estudo de outros meios de acesso, como fibra ótica.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[ARC2000] ARC Eletronics. **RS422 balanced differential drivers** 2000. Endereço Eletrônico: http://www.arcelect.com/rs422.htm. Data da consulta: 00/00/2000.

[COT2000] COTTER, Mark, MCGILVREAY, John. Hirschmann**. Network Systems - High Availability Industrial Automation Networks** 2000. Endereço Eletrônico: http://www.hirschmann-usa.com/Resiliency.htm. Data da consulta: 05/07/2000.

[DAV1997] DAVIS, Harold. **Delphi - ferramentas poderosas**. São Paulo : Berkeley Brasil, 1997.

[DEL2000] DELGADO, Joaquim Duarte Barroca. **Automação industrial – tecnologia CIM (fatores críticos na sua implementação)** 2000. Endereço Eletrônico: http://www.ipv.pt/millenium/arq9_2.htm . Data da consulta: 00/00/2000.

[FAR1998] FARACO, Rafael Avila. **Uma arquitetura de agentes para negociação dentro do domínio do comércio eletrônico** 2000. Endereço Eletrônico: http://www.eps.ufsc.br/disserta98/faraco/index.htm. Data da consulta: 05/07/2000.

[FRA2000] FRANCO, Lúcia Regina Horta R.. **GAII – grupo de automação e informática industrial** 2000. Endereço Eletrônico: http://www.iee.efei.br/~gaii/rs485/hp_rs485.htm#RS-485. Data da consulta: 00/00/2000.

[INM1992] INMETRO - Instituto de Metrologia. **Portaria 074** 1992. Endereço Eletrônico: http://www.inmetro.gov.br. Data da consulta: 05/07/2000.

[KLI1999] KLITZKE, Marcelo. **Protótipo de hardware para aquisição e transmissão de imagens via padrão serial RS485**. Blumenau, 1999. Monografia (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

[LIM2000] LIMA, Maria de Fátima Webber do Prado. **Arquitetura MAP e protocolo MMS** 2000. Endereço Eletrônico: http://penta.ufrgs.br/hometipo.html . Data da consulta: 00/00/2000.

[LON1997] LONGO, Maurício B. **Delphi 3 total - dominando a ferramenta**. Rio de Janeiro : Brasport, 1997.

[MON2000] MONDADA, Matteo. **Fieldbus in industry: what is at stake?** 2000. Endereço Eletrônico: http://vigna.cimsi.cim.ch/tai/BDC/in/BDC.html. Data da consulta: 00/00/2000.

[MOR2000] MORAES, Alan. **Supervisão a distância de um processo automatizado** 2000. Endereço Eletrônico: http://www.instii.fr/projets/platform/Elorn/MATERIEL/MODBUS/pag_01.html . Data da consulta: 00/00/2000.

[OLI1987] OLIVEIRA, Luís Antônio Alves de. **Comunicação de dados e teleprocessamento: uma abordagem básica**. 2. Ed. São Paulo : Atlas, 1987.

[RIB2000] RIBEIRO, Júlio César. **Sistemas de automação industrial** 2000. Endereço Eletrônico: http://www.inhouse.com.br/artigo.html. Data da consulta: 00/00/2000.

[ROG2000] ROGANA, Wagner Gomes, BRACARENSE, Alexandre Queiroz. **Automação industrial** 2000. Endereço Eletrônico: http://www.demec.ufmg.br/Grupos/Solda/Automacao/Index.html#Contrp. Data da consulta: 00/00/2000.

[ROM2000] ROMBALDI, Valéria. **Topologias de redes** 2000. Endereço Eletrônico: http://penta.ufrgs.br/Valeria/topo.html. Data da consulta: 00/00/2000.

[SCH2000] Schneider Eletronics. **Modbus Protocol** 2000. Endereço Eletrônico: http://www.modicon.com/techpubs/toc7.html. Data da consulta: 00/00/2000.

[SIL2000] SILVEIRA FILHO, Elmo Dutra da. **Redes de comunicação de baixo nível (fieldbus)** 2000. Endereço Eletrônico: http://www.malbanet.com.br/professorelmo/Conetfld.htm. Data da consulta: 00/00/2000.

[SOA1993] SOARES NETO, Vicente Soares. **Comunicação de dados: conceitos fundamentais**. São Paulo : Érica, 1993.

[TAN1994] TANENBAUM, Andrew S.. **Redes de computadores**. 2. Ed. Rio de Janeiro : Campus, 1994.

[URB1993] URBANO, Luiz Osvaldo. **Manual da Check 5P**. Ascurra : Casa do Papel, 1993.