UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

CAROLINA PUGGINA LIMA

# Desenvolvimento de um *Handheld* para dispositivos *WirelessHART*

Trabalho de Graduação.

Orientador
Prof. Dr. João Cesar Netto

Porto Alegre, novembro de 2011.

# Agradecimentos

Dedico este trabalho à minha família, que sempre me apoiou e incentivou a persistir na minha jornada de aprendizagem.

Agradeço ao meu namorado Frederico, que esteve ao meu lado em três dos cinco anos que frequentei a universidade. Sua motivação foi fundamental em momentos em que a autoconfiança e a objetividade estavam enfraquecidas.

Realizo este trabalho com esforço e dedicação de todo grupo do LASCAR, que sempre pacientes, ajudaram com suas experiências e seus conhecimentos. Agradeço especialmente ao professor João Cesar Netto, que não só me deu a honra de ser sua orientanda, mas que contribuiu da escolha do tema do trabalho até os últimos testes de laboratório.

Lembro ainda de Deus, que tornou possível mais este grande feito em minha jornada.

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ACK | *Acknowledge* |
| CRC | *Cyclic Redundancy Check* |
| C8PSK | *8 Channel Phase Shift Key* |
| DD | *Device Description* |
| DDL | *Device Description Language* |
| FCS | *Frame Check Sequence* |
| FSK | *Frequency Shift Key* |
| HCF | *HART Communication Foundation* |
| HW | *HART Wired* |
| ID | *Identifier* |
| IEC | *International Eletrotechnical Commission* |
| LASCAR | Laboratório de Sistemas de Controle, Automação e Robótica |
| TDMA | *Time Division Multiple Access* |
| UFRGS | Universidade Federal do Rio Grande do Sul |
| WH | *WirelessHART* |

# LISTA DE FIGURAS

# RESUMO

Os avanços na área de eletrônica vêm permitindo o crescimento tecnológico dos equipamentos de automação para processos industriais. A implantação de uma rede sem fio nas indústrias, além de reduzir o custo de instalação dos dispositivos, possibilita a expansão da rede provendo comunicação com dispositivos em locais de difícil acesso para os cabos e torna as instalações mais seguras ao diminuir a quantidade de cabos elétricos que circulam pela planta.

O trabalho aqui proposto consiste na implementação de um *Handheld* para dispositivos *WirelessHART*, o primeiro padrão internacional de comunicação sem fio para processos industriais. *Handheld* neste contexto é o nome dado ao dispositivo móvel responsável pala manutenção e comissionamento dos dispositivos de rede via porta de comunicação serial, conforme estabelecido por norma.

**Palavras-Chave:** *WirelessHART*, Rede de sensores sem fio, *Handheld*.

# ABSTRACT

Advances in electronics technology have allowed the growth of automation equipment for industrial processes. The deployment of a wireless network industries reduce the cost of installation of the devices, allows the expansion of the network providing communication with devices in places of difficult access to cables and makes the network safer by reducing the amount of electrical cables in the plant

This investigation presents the implementation of a *Handheld* to *WirelessHART* devices, the first international standard of wireless communication for industrial processes. *Handheld* im this context is the name given to the mobile device responsible for the commissioning and maintenance of network devices through serial communication port, as established by norm.

**Keywords:** Wireless*HART*, Wireless sensor network, Handheld

# 1 INTRODUÇÃO

Os avanços na área de eletrônica vêm permitindo o crescimento tecnológico dos equipamentos de automação para processos industriais. Equipamentos cada vez mais robustos e com maior poder computacional tornaram possível a comunicação entre os diversos dispositivos que compõem a planta industrial para um maior controle sobre os processos neste cenário.

Melhorias na tecnologia de sensores levaram ao surgimento de uma grande variedade de dispositivos inteligentes, dotados de microprocessadores e que fornecem uma boa visão dos processos nas indústrias. Quando conectados, estes dispositivos tem o poder de melhorar o desempenho operacional da planta monitorando o funcionamento de maquinários e até mesmo ajustando suas configurações, se necessário.

A implantação de uma rede sem fio nas indústrias, além de reduzir o custo de instalação dos dispositivos[1], possibilita a expansão da rede provendo comunicação com dispositivos em locais de difícil acesso para os cabos, além de tornar as instalações mais seguras ao diminuir a quantidade de cabos elétricos que circulam pela planta. Cada vez mais indústrias estão adotando redes sem fio em suas plantas, com isso a necessidade de desenvolver dispositivos que para manutenção, programação e monitoramento dos equipamentos da rede sem prejudicar a mesma torna-se eminente.

Este trabalho de graduação versará sobre a construção de um *Handheld* para redes *WirelessHART*, um equipamento feito para o comissionamento de dispositivos que utilizam este protocolo. Este *Handheld* serve para programar e monitorar dispositivos de campo *WirelessHART* conectando-se diretamente no mesmo. Entre as utilidades, está a de realizar a manutenção do dispositivo sem sobrecarregar a rede com comandos, obter respostas sobre o estado do dispositivo mais rapidamente do via gerenciador de rede pelo *Gateway*, possibilitar a programação dos dispositivos e das redes modificando informações cruciais nos dispositivos, como a *Join Key* e a *Network ID* e ainda monitorar a rede a partir de dados colhidos de algum dispositivo.

## 1.1 Contextualização

Desde a década de oitenta existe grande esforço em projetar e especificar protocolos de comunicação que tornem mais seguro o controle de processos industriais. Algumas organizações têm promovido o uso de tecnologias wireless nas industrias, dentre elas *Bluetooth, WiFi, Zigbee* e *Wina* [10], mas nenhuma obteve sucesso em estabelecer um padrão absoluto para as indústrias, principalmente pela falta de robustez e segurança[2].

O ambiente industrial é extremamente hostil. Pode apresentar ruído eletromagnético de grande intensidade em alguns momentos, como no acionamento de motores elétricos. Estes ambientes também costumam apresentar temperatura e umidade elevadas, aspectos que costumam ser prejudiciais aos componentes utilizados em sistemas computacionais e

de comunicação. Neste contexto, os equipamentos designados para o uso em indústrias são especialmente construídos para atuar nestas condições adversas, assim como os protocolos de comunicação adotados também devem considerar aspectos de segurança e disponibilidade nestas condições[5].

À frente dos demais, a *HART* Communication Foundation (HCF) [3] lançou em 2007 a norma *HART* 7[4], que já contava com as seções relativas ao *WirelessHART*, sendo este portanto o primeiro padrão aberto de comunicação sem fio especificamente desenvolvido para uso industrial. Desde então, a venda de equipamentos que utilizam este protocolo está em ascensão, o que indica que o protocolo está tendo uma boa aceitação no mercado. Isto deve-se, principalmente, ao fato do protocolo ser robusto e seguro. Em 2010, o protocolo *WirelessHART* foi reconhecido pela IEC como primeiro padrão completo internacional de comunicação sem fio para processos na indústria [7].

Uma vez feita a instalação da rede *WirelessHART* na fábrica surge o interesse em técnicas de manutenção e programação dos dispositivos desta rede. A manutenção e a programação dos dispositivos, de acordo com o padrão *WirelessHART*, devem ser feitas por um equipamento específico para tal, via porta serial localizada no próprio dispositivo, chamada de porta de manutenção. O trabalho aqui proposto seria de criar uma ferramenta para atuar na porta de manutenção dos dispositivos, possibilitando que qualquer usuário pudesse configurar qualquer dispositivo, inclusive recém vindos de fábrica, para entrarem na rede, assim como modificar dados de dispositivos necessários para a continuidade dos processos industriais.

## 1.2 Motivação

Um projeto visando à implementação de uma rede de sensores sem fio para medir e controlar abertura de válvulas está sendo desenvolvido no LASCAR (Laboratório de Sistemas de Controle e Automação) da UFRGS em parceria com a Petrobrás. Os dispositivos de rede deste projeto já foram feitos, tanto as placas de hardware como os firmwares, e estes se encontram em etapa de testes.

Cada dispositivo *WirelessHART* vem com suas próprias configurações de fábrica. Para poder conectar os dispositivos formando uma rede é necessário configurá-los e comissioná-los. É imprescindível um software para estas configurações e comissionamentos, tanto para formar redes adequadamente quanto para colher dados para monitoramento. O *Handheld* é exatamente este software, tão necessário para estudos e implementação de redes em indústrias.

Um software que atue como um *Handheld* nestes dispositivos é imprescindível, primariamente configurando estes dispositivos para formarem as redes adequadamente e posteriormente para monitorar dados da rede e dos próprios dispositivos.

Inicialmente o projeto consistia em criar um *Handheld* que se comunicasse com os dispositivos via comunicação wireless, mas para tal o gerenciador de rede precisaria aprovar a comunicação do *Handheld* com o dispositivo criando um *superframe* especial para essa comunicação. Como este *Handheld* seria uma inovação no mercado visto que hoje não existem equipamentos a venda com esta opção de comunicação para redes *WirelessHART*, os gerenciadores de rede disponíveis no mercado não suportam estes comandos. No caso de *Gateway*s que possuem gerenciadores de rede integrados, o *Gateway* nada responde ao ser solicitado o comando específico para criação do *superframe* especial para *Handheld*, embora este comando esteja previstos em norma.

Desta forma, embora o *Handheld* desenvolvido comunique-se com os dispositivos via uma porta serial e não por comunicação sem fio, ainda sim o projeto é fundamental para o comissionamento dos dispositivos em uma rede industrial. O projeto assim se caracteriza por um *Handheld* visto que pode ser portado para um pequeno equipamento, como um *tablet*, e assim configurar cada válvula da planta, sendo está já instalada ou não.

O estudo sobre o protocolo, o encapsulamento e o desencapsulamento dos pacotes assim como o estudo sobre as interpretações das respostas foram exaustivas, garantindo a segurança na comunicação do *Handheld* com os dispositivos.

# 2 COMPONENTES DE UMA INSTALAÇÃO WH

Uma rede *WirelessHART* suporta uma grande variedade de dispositivos de diferentes fabricantes. A Figura 1 ilustra os elementos básicos de uma instalação e os tipos mais comuns de equipamentos que podem estar presentes. Em alguns casos, como em um Field Device, o produto é um dispositivo físico. Em outros casos, como o Network Manager, o produto é um elemento lógico, sendo descrito abstratamente.

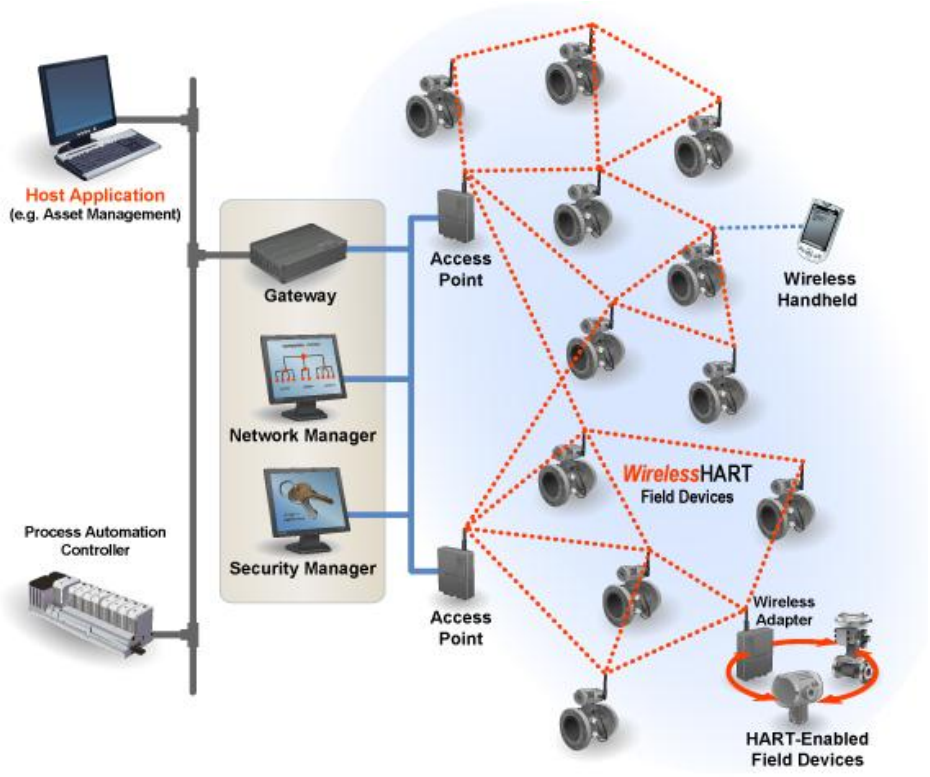



**Figura 1. Elementos de uma instalação *WirelessHART*.**

## 2.1 Dispositivos de Rede

Dispositivo de rede (*Network Device*) é o nome dado para aqueles dispositivos que atuam como nós em uma rede. Em uma planta industrial eles são os dispositivos diretamente ligados à planta industrial, monitorando e interagindo nos processos industriais.

Cada dispositivo de rede possui um endereço único (*Unique Adress*) para ser identificado na rede e uma lista de vizinhos, identificados em operações de escuta da rede.

Estes vizinhos podem ser identificados em qualquer slot, ou seja, há qualquer momento. Os dispositivos de rede mais comuns são os dispositivos de campo, adaptadores e *Gateway*s.

Todo dispositivo de rede deve ser capaz de rotear pacotes e de entregar pacotes em nome de outro dispositivo da rede. Uma tabela interna do dispositivo é usada para decidir o roteamento que será dado aos pacotes.

**Dispositivos de Campo**

Os dispositivos de campo (*Field Devices*) são os que aparecem na figura nomeados por letras de '*a*' a '*n*'. Este é o tipo mais comum de dispositivo de rede, sendo eles os dispositivos que captam informações do maquinário e do ambiente e transmitem as informações pertinentes pela rede, tornando possível o controle de processos. No projeto em questão, os dispositivos de campo são rádios ligados a sensores que monitoram o status, a abertura e o fechamento de válvulas. A Figura 2 a seguir mostra este dispositivo.

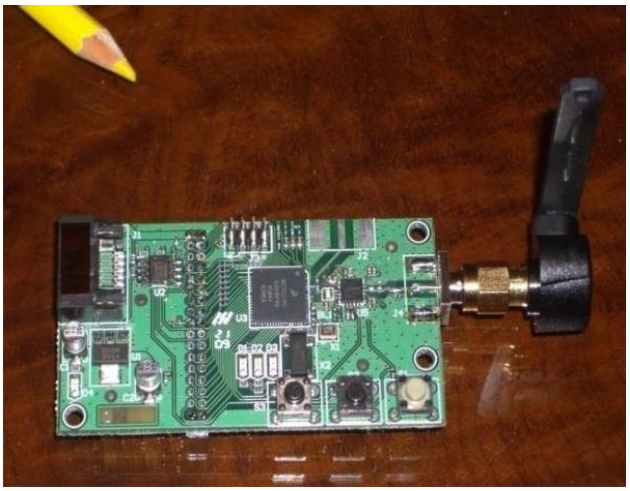


**Figura 2. Exemplo de dispositivo de campo *WirelessHART*.**

**Adaptador WH**

O adaptador *WirelessHART* é um dispositivo de rede que adiciona a comunicação *WirelessHART* a um dispositivo que já seja um *HART Field Device*, ou seja, já possua a comunicação *HART* Wired. Ele atua como um tradutor entre *HART Wired* e *WirelessHART*, permitindo ao dispositivo de campo transmitir e receber mensagens em qualquer um dos protocolos. Este adaptador é mostrado na Figura 3.


**Figura 3. Exemplo de adaptador *HART - WirelessHART*.**

***Gateway***

*Gateway* é o dispositivo que provê ponto de acesso para a rede, sendo o intermediário entre a rede de sensores e a planta de automação. Pode haver mais de um ponto de acesso por rede ou ainda mais de uma rede com o mesmo ponto de acesso, desde que cada rede tenha ao menos um *Gateway*. O *Gateway* conecta a rede *WirelessHART* à planta de automação, possibilitando a troca de dados entre redes. Os dados coletados pelo *Gateway* são passados para a planta de automação via mensagens que podem ser:

- mensagens de rotina, comunicando periodicamente dados de processos e eventos;
- mensagem de status, quando um dispositivo de campo apresentar condição de processo não recomendável ou falha de funcionamento;
- mensagens de configuração e manutenção da rede, geralmente passados por bursts.

**Figura 4. Exemplo de *Gateway WirelessHART*.**

As aplicações que controlam os processos industriais precisam do *Gateway* para se comunicar, monitorar e configurar os demais dispositivos de rede. O *Gateway* também provê o relógio global da rede para os seus dispositivos. Um exemplo de *Gateway* é mostrado na Figura 4, é um modelo da Emerson que já possui ponto de acesso e

gerenciador de rede integrados como forma de garantir que a comunicação entre estes dispositivos sempre será segura.

## 2.2. Gerenciador de Rede

O gerenciador de rede (*Network Manager*) é uma aplicação que gerencia a rede *WirelessHART*. Para cada rede deve haver um e apenas um gerenciador de rede. Ele é o responsável por formar a rede, controlar a entrada de novos dispositivos na rede *(joins)*, configurar os dispositivos de rede e monitorar a saúde da rede.

O gerenciador de rede possui uma lista de todos dispositivos de rede e garante que cada um deles terá um *nickname* único de 16 bits. Esta lista é usada para gerar funções de rede, como a função de roteamento e de agendar comunicação entre os dispositivos de rede. O gerenciador de rede também gerencia *routing tables*, para monitor a saúde da rede coletando dados de desempenho e diagnóstico. Todas as informações são analisadas em tempo real, permitindo a reconfiguração da rede com a mesma em funcionamento quando detectado algum problema.

Este tipo de dispositivo deve ter uma rota segura ao *Gateway* da rede, pois sem a comunicação entre eles é inviável manter a rede *WirelessHART*. Nos *Gateway*s utilizados no projeto em estudo, o *network manager* já vem integrado com o *Gateway*, sendo adquirido direto do fabricante como um único produto, como é o caso do *Gateway* usado como exemplo na figura 4, garantindo esta rota segura entre *Gateway* e *network manager*.

## 2.3 *Handheld*

*Handheld* representa a futura geração de instrumentos para redes *WirelessHART*. É pouco utilizado atualmente, mas deve passar a ser cada vez mais presente com o aumento da utilização das redes *WirelessHART*. Este dispositivo serve para manutenção dos dispositivos de campo e pode estar em um computador portátil, em um dispositivo de campo reconfigurado ou ainda em *Palmtops, Tablets* e celulares. Ou seja, *Handheld* é um computador portátil com uma aplicação host e conexão para a configuração de outros dispositivos da rede.

Este dispositivo portátil tem acesso aos dados dos processos enquanto tem o poder de gerenciar, atualizar e configurar os dispositivos de rede, além de programar dispositivos para entrar nas redes. A Figura 5 mostra um *Handheld* disponível no mercado atualmente, que assim como o *Handheld* do projeto, comunica-se com os dispositivos via porta de manutenção, ou seja, via fios [9].


**Figura 5. *Handheld* da Emerson para dispositivos *WirelessHART*.**

# 3 PROTOCOLOS *HART* E *WIRELESSHART*

O *Handheld* atua em uma rede de dispositivos que enviam e recebem comandos utilizando o protocolo *WirelessHART*, mas se comunicará com os mesmos via cabo. A comunicação nestes dispositivos, de acordo com a norma[4], dá-se por protocolo *HART* quando for feita via fios e por *WirelessHART* quando via comunicação sem fios, logo, a comunicação *Handheld*-dispositivo é estabelecida via protocolo *HART*. A explicação para o mesmo dispositivo poder estabelecer comunicação por dois protocolos diferentes, de acordo com meio físico da transmissão, é a compatibilidade entre os dois protocolos.

Ao criar o protocolo *WirelessHART* a organização *HART* teve o intuito de deixa-lo totalmente compatível com o protocolo *HART* Wired, para viabilizar que dispositivos *HART* e *WirelessHART* convivam em uma mesma rede. Portanto, o protocolo *WirelessHART* manteve o mesmo formato de *frame* do *HART*, assim como seguiu as especificações dos comandos *HART* já existentes.

As comunicações *WirelessHART* e *HART* são baseadas em modelo de comunicação estilo mestre-escravo, em que o dispositivo recebe um comando no formato do *frame* de comunicação, executa o que foi solicitado e em seguida responde à solicitação. Assim, a compatibilidade está mantida uma vez que os dispositivos *WirelessHART* mantém o formato de *frame* de comunicação e especificação dos comandos do protocolo *HART*.

## 3.1 *Frame HART*

O *frame HART* Wired, ou simplesmente *HART*, é o que define o formato e o conteúdo das mensagens a serem trocadas entre dispositivos da rede. Cada comando possui um código correspondente, que identifica o comando, sendo este o número do comando.

O *frame* foi desenvolvido inicialmente com apenas um byte destinado ao envio do número do comando, considerando assim que a numeração de 0 a 255 seria suficiente para identificar todos os comandos possíveis. A Figura 6 a seguir mostra cada campo do formato deste *frame*.


**Figura 6. *Frame* de troca de mensagens *HART***

Os campos *Byte Count* e *Check Byte* servem para verificar a integridade da mensagem. O *Check Byte* é um *CRC* calculado com todos os demais bytes da mensagem e o *Byte Count* diz quantos bytes terá o campo *Data* na mensagem em questão, assim o dispositivo sabe quantos bytes ainda terá a mensagem até que chegue o *CRC*. O campo *Delimiter* contém o símbolo indicando que a mensagem irá começar, sendo este o primeiro byte

após o preâmbulo, que são alguns bytes em 0xFF (hexadecimal) utilizados para a sincronização de mensagens no meio sem fio. Em *Command* está o número do comando, de 0 a 255.

No campo *Address* é passado o endereço do dispositivo alvo da mensagem. No *Handheld,* primeira mensagem, solicitando conexão, é passada com endereço 1. Na resposta dessa primeira mensagem o dispositivo envia seu *Unique ID,* o valor único do dispositivo, e as próximas mensagens terão como *Adress* os cinco bytes do *Unique ID.* O campo *Expansion* é reservado pela norma e ainda não utilizado. Pode variar de 0 a 3 bytes, e como ainda não está sendo utilizado pela *HCF*, na prática é um campo de zero bytes, que não aparece no *frame*.

O *Handheld*, ao receber o primeiro byte, confere se este faz parte do preâmbulo. Caso afirmativo, descarta os bytes em 0xFF e aguarda receber o *Delimiter* 0x06 e mais sete bytes, cinco de *Adress* ,um de *Command* e o último de *Byte Count*. O último byte recebido, o *Byte Count*, diz quantos bytes ainda serão recebidos até o final da mensagem, e este valor será o novo valor de bytes esperados.

Quando a numeração de 0 a 255 já não foi mais suficiente para expressar todos os comandos possíveis, foram criados então novos comandos com numeração superior a 255. Para que todos dispositivos, *WirelessHART* e *HART*, suportem todos os comandos, foi criado um comando especial para indicar que o número do comando que será solicitado é maior que 255, ou seja, de 16 bits.

Este comando funciona como uma *flag.* Ao colocar o valor 31 (0x1F hexadecimal) no campo *Command*, simbolizando que o número do comando é estendido, o valor do comando a ser executado deverá ser representado por dois bytes no início do campo *Data*. Como o campo *Command* é de apenas um byte, a solução encontrada para escrever comandos de 16 bits foi acrescentar dois bytes no campo *Data,* aproveitando que este campo é de tamanho variável.

**Solicitação com comandos em um único byte**

Para os comandos de numeração até 30 ou entre o 32 e o 253, o número do comando aparece no campo *Command.* Com apenas o número do comando, o dispositivo escravo sabe o que esperar de parâmetros para este comando e os busca, estando estes presentes a partir do primeiro byte (byte zero) do campo *Data* .

O dispositivo escravo, caso execute o comando sem perceber erros, retornará a mensagem com o número do comando executado no campo *Command* e com o status nos dois primeiros bytes de *Data,* antes da resposta do comando. Se houver algum problema na execução do comando, como um aviso ao dispositivo, no campo *Command* não terá o número do comando, mas um outro número que simbolize código de erro.

**Solicitação com comandos estendidos**

A mensagem de comando estendido apresenta o número 31 no campo *Command* e o número do comando nos dois primeiros bytes do campo *Data.* Assim, se um dispositivo receber uma mensagem com a *flag* de comando estendido e menos de dois bytes no campo *Byte Count* deve responder com mensagem de erro "*too few data bytes*".

Toda resposta de comando estendido deve apresentar no campo *Command* o número 31 e, antes dos bytes de resposta propriamente ditos, os dois bytes de status seguidos dos dois bytes com o número do comando executado no campo *Data*. Se o comando for

executado corretamente, a resposta deve ter o número 31 no campo *Command* e o número do comando nos terceiro e quarto bytes de *Data*. [6]

## 3.2 *WirelessHART*

Os protocolos *HART* e *WirelessHART* implementam a comunicação do meio físico até a camada de aplicação. Uma tabela comparando os protocolos *HART* com o modelo de sete camadas OSI pode ser vista na Figura 7.

| Camada OSI | Função | HART | |
|---|---|---|---|
| 7 Aplicação | Fornece aos usuários as aplicações da rede. | Comandos orientados. Tipos de dados pré-definidos. Procedimentos de aplicações. | |
| 6 Apresentação | Converte dados de aplicação entre a rede e a máquina local | | |
| 5 Sessão | Conecta os serviços de gerenciamento com as aplicações. | | |
| 4 Transporte | Transfere mensagens de forma transparente e independente na rede. | Transferência de conjunto de dados. Segmentação de dados. Negociação da segmentação de dados. | |
| 3 Rede | Roteamento dos pacotes de ponta a ponta. Endereçamento da rede. | | Otimização de energia, redundância de caminhos. Auto organização da rede. |
| 2 Enlace | Estabelece pacote da estrutura de dados. Detecção de erros. | Controle de recebimento dos dados dentro da estrutura de pacotes. | Segurança e confiabilidade. Tempo de sincronização. TDMA/CSMA. |
| 1 Física | Conexão Elétrica / mecânica e transmissão. | Sinal analógico e digital simultaneamente. | Wireless 2.4 GHz, baseado em rádios 802.15.4, 10dBm. |
| | | Protocolo HART Wired | Protocolo WirelessHART |

**Figura 7. Tabela de comparação modelo OSI vs protocolos *HART*.**

O protocolo *WirelessHART* define o meio físico, descrevendo que os dispositivos de campo devem ser rádios de 2,4GHz, seguindo o padrão 802.15.4, com potência de transmissão de 10dBm. O protocolo define ainda o enlace, que garante segurança e confiabilidade na transmissão dos dados baseando-se em TDMA e saltos de frequência. O TDMA (*Time Division Multiple Access*) é uma tecnologia de transmissão digital que permite que vários usuários utilizem o mesmo canal de comunicação pela divisão do tempo de utilização deste canal.

As comunicações neste protocolo ocorrem em *time slot*, ou fatias de tempo, de 10ms. Cada *time slot*, pode ser associado a um *link*, uma conexão física entre dois *peers*. Assim, múltiplos nós da rede compartilham o mesmo canal de comunicação utilizando somente uma parte da banda passante [6]. Caso o *time slot* seja associado a um *link*, o meio

utilizado será TDMA. Caso o *time slot* ainda não esteja associado a nenhum *link*, será utilizado CSMA/CA[11].

A coleção de todos os *time slots* repetindo no tempo formam um *superframe*. Toda comunicação *WirelessHART* é feita dentro destes *superframes*. O número de *slots* de um *superframe* determina quão frequente o *slos* se repetirá. Quando um *superframe* é criado, no gerenciador de rede é gerada uma tabela que guarda as informações sobre as associações dos links aos slots. Na Figura 8 está ilustrado um *superframe* com 3 *time slots*. Cada vez que os 3 *slots* se repetem é caracterizado um ciclo. Na figura, os slots 1 e 2 possuem *links*, enquanto o *slot* 3 ainda está vazio. Podemos verificar que o primeiro *slot* está vinculado ao *link* que está sendo utilizado para a comunicação do dispositivo A com o B, e que ao segundo *slot* está vinculado um *link* de comunicação de B com C [6].

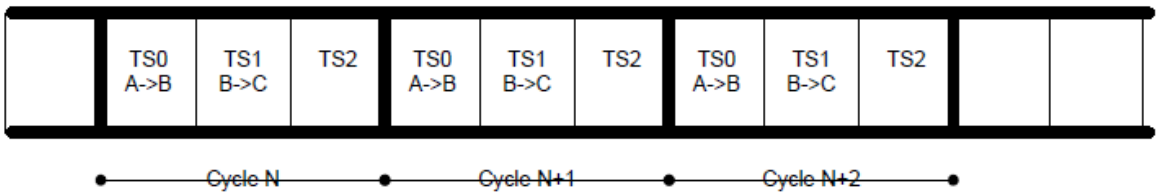



**Figura 8. Exemplo de um *Superframe* com 3 *time slots*.**

Na camada de rede do modelo OSI é feita a otimização de energia, procurando a auto organização da rede a fim de não comprometer a carga da bateria dos nós da rede. Esta preocupação é coerente visto que os dispositivos de campo *WirelessHART* geralmente são alimentados por uma bateria, pois mesmo que no local seja possível instalar cabeamentos, no ambiente industrial é mais seguro que o dispositivo seja alimentado por uma bateria interna do que por meio de cabos, que podem gerar curto-circuito ou pegar fogo no caso de algum maquinário soltar alguma faísca.

Desta forma, além do protocolo *WirelessHART* ser um protocolo que visa prover uma comunicação confiável, também precisa prover uma comunicação que propicie baixo consumo. Para conseguir o baixo consumo foram pesquisadas técnicas de espalhamento espectral, originalmente desenvolvidas para uso militar. Essas técnicas permitem redução de energia por bit, redução de distorções e interferências, além de dificultar a detecção da mesma. Tipicamente nestes sistemas, o espalhamento espectral é obtido por saltos de frequência.

Assim, os *links* disponibilizados pelo *Gerenciador de Rede* para comunicação dos dispositivos não serão com a mesma frequência, mas a sequência de saltos de canal é conhecida pelo receptor que calcula a próxima frequência utilizando a semente do código pseudoaleatório. Assim os dispositivos sempre calculam em que frequência estabelecerão a próxima comunicação com o *Gateway*.

Existem *superframe* especiais, como o *superframe* destinado para a comunicação de um *Handheld* com um dispositivo de campo. Para ativar este *superframe* o *Handheld* deve solicitar ao *Gateway* que ative a comunicação especial nos dispositivos de campo desejados. Os comandos 806 e 807 foram feitos exclusivamente para esta comunicação entre *Handheld* e dispositivos de campo, mas eles falharam ao ser enviados aos *Gateways* disponíveis no mercado durante o desenvolvimento deste trabalho de graduação. Por este

motivo a comunicação sem fios entre o *Handheld* e os dispositivos de campo não foi possível.

Contudo, segundo o diz as especificações dos protocolos, quando os dispositivos de campo têm a sua porta de manutenção conectada, estes automaticamente passam a responder às solicitações desta, assim como respondem às solicitações do *Gateway*. Esta foi a forma encontrada para estabelecer conexão do *Handheld* com os dispositivos de campo, através das portas de manutenção dos mesmos.

# 4 DESCRIÇÃO DO HARDWARE DO PROJETO

O *Handheld* desenvolvido fez parte do projeto do grupo LASCAR da UFRGS em parceria com a Petrobrás para o desenvolvimento de dispositivos de uma rede sem fio para atuar em válvulas da planta industrial da Petrobras. Estes dispositivos devem ajudar nos processos industriais, medindo o grau de abertura de cada válvula assim como recebendo comandos que ditem que a válvula deve ter um novo grau de abertura e atuando nas válvulas para estabelecer o novo grau de abertura solicitado. Os dispositivos deste projeto são, portanto, hardwares que funcionarão como rádios acoplados às válvulas industriais e que formarão uma rede sem fio entre eles.

A escolha do protocolo *WirelessHART* para esta rede sem fio, como explicado anteriormente, foi devida a grande aceitação que este protocolo vem apresentando na área industrial, sendo um protocolo considerado robusto, de baixo consumo de potência e seguro. Segundo a norma, para o dispositivo ser *WirelessHART* compatível precisa ser também *HART* Wired compatível e possuir uma conexão *HART* Wired para configurações e manutenção, a porta de manutenção, citada anteriormente.

Os rádios foram montados a partir de um kit da Freescale, o "platform-in-package" Freescale MC 13224. Este chip da Freescale já vem com um microcontrolador ARM 7 e um radio IEEE 802.15.4 incluso. Esta placa foi um bom ponto de partida, embora não suprisse ainda todas as necessidades do projeto. Outros módulos então foram acoplados nos rádios, como um amplificador (Texas Instruments CC2591) que amplia a sensibilidade de recepção em 10 dB e aumenta 20 dB na transmissão, conexão de programação e debug JTAG e uma porta RS-485 para a conexão da porta de manutenção. Com uma potência de zero dBm na saída do MC 13224 temos uma potência de 100mW(20dBm) na base da antena. A placa desenvolvida pode ser observada na Figura 9 a seguir.

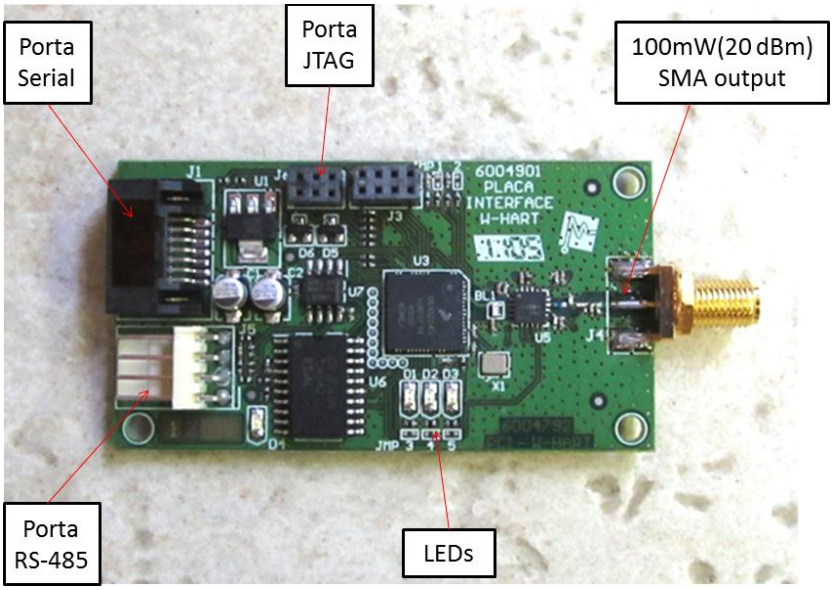



**Figura 9. Placa de hardware desenvolvida no projeto.**

A conexão do rádio com a válvula é feita por porta I2C, que liga o rádio ao atuador da válvula por um conector SATA. Foi adquirido um firmware da STG para que este rádio aceite e responda aos comandos enviados tanto pela comunicação wireless quanto por fios, mas este firmware 'genérico' também teve que ser modificado para que atendesse às necessidades do projeto. Os próprios membros do projeto modificaram e posteriormente testaram o firmware para que enviasse e recebesse dados pela I2C, comunicando-se assim com o atuador da válvula.

A ligação do rádio com o *Handheld* precisa ser feita de acordo com o padrão, por algum meio físico aceito na norma *HART* [4], que diz que a ligação pode ser feita por conexão FSK, C8PSK ou RS-485. No projeto optamos pela comunicação RS-485, por ser a conexão de menor custo e pela facilidade com a qual encontramos adaptador para porta USB do computador. Foi utilizado um adaptador Novus i485 para conectar a porta RS-485 ao computador via porta USB. Esta ligação pode ser vista na Figura 10. Maiores detalhes podem ser vistos em [8].

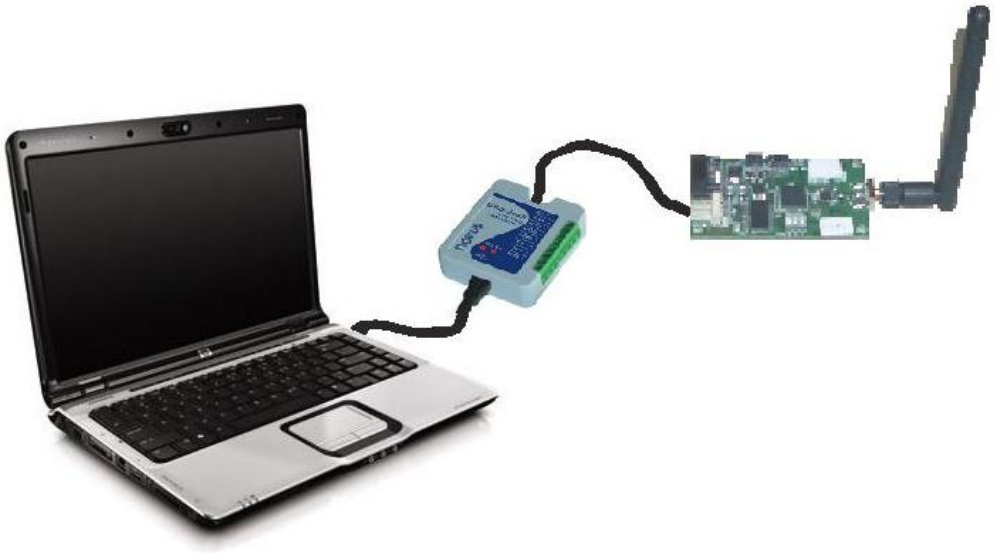


**Figura 10. Ligação do computador com o rádio via adaptador Novus i485.**

Para o *Handheld* é transparente qual o tipo de cabeamento utilizado para a comunicação com os dispositivos. Por este motivo não serão especificadas as propriedades de cada uma das opções de conexões, mas apenas salienta-se que assim como a porta escolhida no projeto foi a RS-485, poderia ter sido a FKS ou a C8PSK, mas estas outras precisariam de moduladores instalados nos dois extremos do cabo, aumentando significativamente o custo do dispositivo por unidade.

Este *Handheld* pode ser conectado a qualquer dispositivo *WirelessHART* para o comissionamento do mesmo, basta que a porta de manutenção deste seja conectada a algum adaptador USB. Para o software, qualquer conexão será vista como uma conexão com uma porta serial, como uma porta COM do Windows, sem diferenciar o tipo de conexão física entre o computador e a porta de manutenção do dispositivo.

# 5 IMPLEMENTAÇÃO DO SOFTWARE

O *Handheld* desenvolvido consiste em um aplicativo para ser instalado em um computador portátil, um notebook qualquer ou tablet, que possua o sistema operacional Windows instalado. Este computador será conectado ao dispositivo de campo por uma porta RS-485 via conversor. Esse aplicativo permite ao usuário solicitar comandos para ajustar o maquinário, configurar os dispositivos para entrarem em uma rede, ou ainda colher dados periodicamente a fim de monitoração da rede e dos dispositivos. O aplicativo deve ser de fácil manuseio do usuário, bastando pressionar alguns botões para estabelecer conexão com o dispositivo e solicitar os comandos desejados. O executável gerado no final do projeto é mostrado na Figura 11, sendo todo aplicativo composto por esta janela única, para simplificar a interface com o usuário:

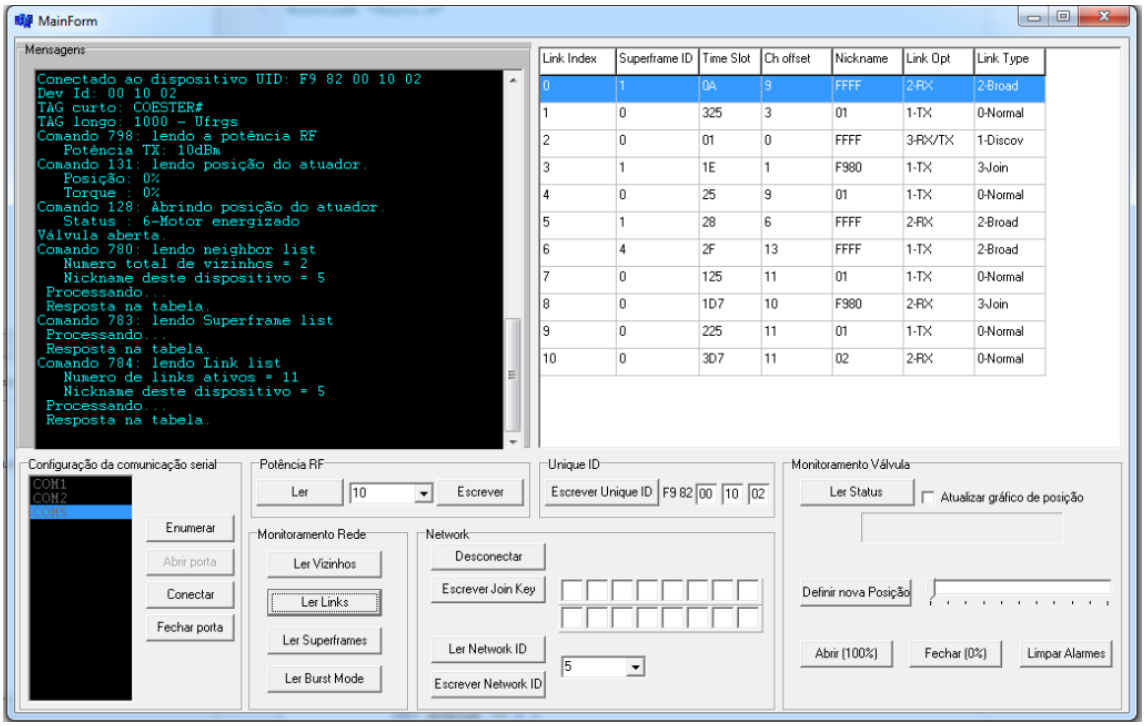



**Figura 11. Aplicativo a ser executado no *Handheld*.**

Como um dos principais fatores considerados na especificação do projeto foi gerar um aplicativo de fácil manuseio, o ambiente de desenvolvimento escolhido para o desenvolvimento deste aplicativo foi o Builder C++ da Embarcadero, pela facilidade com que o programador utiliza *forms* e bibliotecas gráficas e pela simplicidade ao gerar

executáveis livres do compilador ao final do desenvolvimento. Ressalto ainda que o programa desenvolvido não está preso a nenhum ambiente de desenvolvimento, salvo pela plataforma gráfica fornecida pelo Builder da Embarcadero. Modificando a interface com o usuário este mesmo código pode ser portado para qualquer outro ambiente de desenvolvimento C++. Da mesma forma, nada prende o programa à plataforma Windows, salvo a API de programação para comunicação com portas COM padrão utilizada, cujo header incluso nos códigos é o <windows.h> .

A plataforma Windows foi mais estipulada do que escolhida, visto que é este o sistema operacional instalado em todos os computadores do projeto, tanto nos desktops quanto nos notebooks, incluindo os computadores em que será utilizado o executável final. Logo, a compatibilidade com o Windows 7 foi também um dos fatores que influenciou na escolha pelo ambiente de desenvolvimento Builder C++.

## 5.1 Lista de comandos implementados

A lista dos comandos que foram implementados para o software do *Handheld* é apresentada a seguir. O aplicativo futuramente pode ser incrementado de qualquer outro comando sem muito esforço, visto que os protótipos das funções estão prontos e que todos comandos acabam seguindo a mesma linha de implementação e utilizando as funções já prontas para enviar e receber mensagens.

- 13 – Ler *Short Tag*
- 20 – Ler *Long Tag*
- 105 – Ler *Burst mode*
- 124 – Escrever *Unique ID*
- 128 – Abrir Atuador
- 129 – Fechar Atuador
- 130 – Escrever comando para Atuador
- 131 – Ler Status
- 768 – Escrever *Join Key*
- 773 – Escrever *Network ID*
- 774 – Ler *Network ID*
- 780 – Ler lista de Vizinhos do Dispositivo
- 781 – Ler *Nickname* do Dispositivo
- 783 – Ler lista de *Superframes*
- 784 – Ler lista de *Links*
- 797 – Escrever potência RF
- 798 – Ler potência RF
- 960 – Desconectar da rede.

## 5.2 Descrição da Implementação

O projeto todo foi escrito em C++ e possui as funções divididas em diferentes arquivos *.cpp*, as *Units*. A *Unit* chamada *Serial.cpp* contém funções próprias para estabelecer comunicação via porta COM do Windows. A *Unit1.cpp* contém todas as funções do *form* que interagem com o usuário, mas não formula nenhum comando para ser enviado, é apenas um código intermediário. Cada comando é formulado e tem sua resposta interpretada por uma *Unit* específica, cujos nomes são compostos da seguinte maneira: *cmd<nº>.cpp* e *cmd<nº>.h*

A *Unit1.cpp* é o arquivo pelo qual todas as outras *Units* do programa se comunicam, sendo esta a única responsável por chamar funções da *Serial.cpp*, por chamar as funções dos comandos solicitados pelo usuário e por receber e enviar mensagens. É na *Unit1.cpp* que são recebidos os dados, desencapsulados e passados para as *Units* específicas do comando desejado. É nela também que os dados de resposta são encapsulados e enviados para o hardware.

O fluxo de eventos do programa é mostrado na Figura 12. Todo fluxo do programa consiste em uma única *thread*, visto que o programa é orientado a eventos dependentes uns dos outros. A ordem dos eventos é mostrada no diagrama de sequência na figura a seguir.

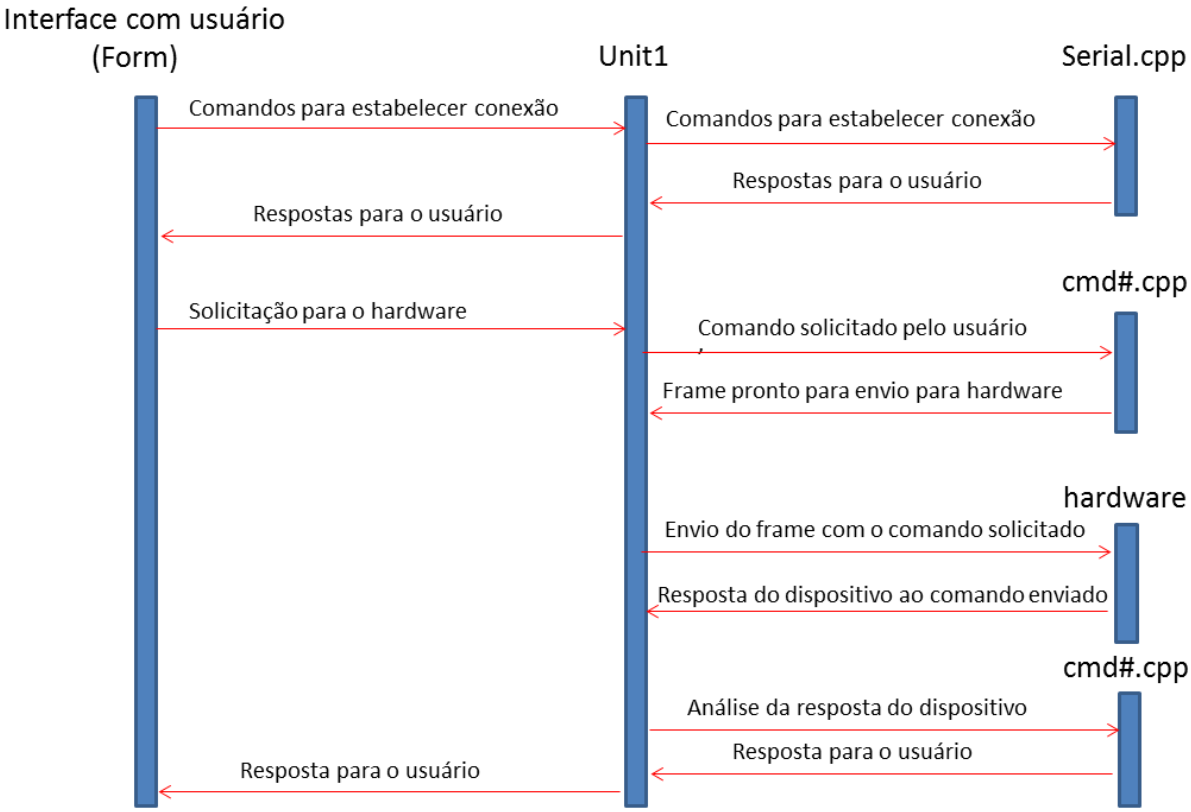



**Figura 12. Fluxo dos eventos no aplicativo.**

## 5.3 Implementação do protocolo *HART* no *Handheld*

As camadas a serem implementadas do protocolo *HART* seriam equivalentes as camadas 1, 2 ,4 e 7 do modelo OSI, conforme pode ser visto na Figura 13. Cada camada precisou ser devidamente tratada pelo *Handheld* a fim da comunicação ser devidamente estabelecida. Neste sessão será descrito como foram implementadas cada uma das camadas, assim como todas as preocupações e cuidados ao transformá-las em código.

| Camada OSI | Função | HART | |
|---|---|---|---|
| 7 Aplicação | Fornece aos usuários as aplicações da rede. | Comandos orientados. Tipos de dados pré-definidos. Procedimentos de aplicações. | |
| 6 Apresentação | Converte dados de aplicação entre a rede e a máquina local | | |
| 5 Sessão | Conecta os serviços de gerenciamento com as aplicações. | | |
| 4 Transporte | Transfere mensagens de forma transparente e independente na rede. | Transferência de conjunto de dados. Segmentação de dados. Negociação da segmentação de dados. | |
| 3 Rede | Roteamento dos pacotes de ponta a ponta. Endereçamento da rede. | | Otimização de energia, redundância de caminhos. Auto organização da rede. |
| 2 Enlace | Estabelece pacote da estrutura de dados. Detecção de erros. | Controle de recebimento dos dados dentro da estrutura de pacotes. | Segurança e confiabilidade. Tempo de sincronização. TDMA/CSMA. |
| 1 Física | Conexão Elétrica / mecânica e transmissão. | Sinal analógico e digital simultaneamente. | Wireless 2.4 GHz, baseado em rádios 802.15.4, 10dBm. |
| | | Protocolo HART Wired | Protocolo WirelessHART |

**Figura 13. Tabela de comparação modelo OSI vs protocolos *HART*.**

## Camada Física

A primeira camada, ou camada física, que no protocolo oficialmente é implementada por cabeamento, o cabeamento utilizado foi o adaptados RS 485 que conecta a porta de manutenção do dispositivo com a porta USB do computador, conforme explicado na última sessão. No projeto em parceria com a Petrobrás, o cabeamento que permite a conexão da porta RS-485 do dispositivo com a porta USB do computador é mostrado na Figura 14, que ilustra um conversor de qualquer entrada possível do dispositivo ligando uma porta de manutenção de um dispositivo de campo *WirelessHART* a um computador portátil.

**Figura 14. Conversor USB conectando um nó da rede ao computador.**

**Camada de Enlace**

A segunda camada do modelo OSI, a camada de enlace, trata do recebimento e transmissão dos dados dentro do formato de pacote estipulado no protocolo em questão. O software do computador portátil ao receber um pacote deve verificar se o preâmbulo está presente assim como se o último byte de dados, o CRC, está coerente com os demais bytes recebidos na transmissão.

Ao enviar pacotes, o software deve ter o mesmo cuidado de encapsular os dados a serem transmitidos na ordem correta, contendo o preâmbulo inicial e o CRC para verificação da integridade dos dados no final.

No programa, a função *waitForResponse*, responsável pela recepção dos dados, ao receber o pacote verifica se os primeiros bytes correspondem ao preâmbulo esperado, e após esta constatação, retira os dados necessários para a aplicação colocando-os em ponteiros globais - buffers. Caso haja erro no preâmbulo, será enviado ao usuário mensagem de erro, e a resposta não será considerada válida.

Para enviar pacotes, existe uma função que é chamada para colocar o preâmbulo e o comando desejado corretamente, a função *buildHeader*. Após a inserção de todos os campos desejados, antes de enviar o comando, outra função ainda é chamada, a *buildFCS*, que faz o cálculo do CRC com todos os demais bytes do pacote para controle de erro do receptor da mensagem.

**Camada de Transporte**

A camada de transporte também está presente na composição do *frame* de comunicação. Todos os dados recebidos e enviados devem estar no local correto dentro do *frame*. A aplicação deve, portanto, conseguir retirar os dados precisos para a execução do software, separando estes do restante do pacote. Esta composição do *frame* dá-se graças a camada de transporte, e o *Handheld* segue este formato de *frame* para receber e enviar dados.

Conhecendo o formato do *frame* de comunicação, já mostrado na sessão 3, podemos desenvolver aplicações que buscam diretamente no *frame* cada campo que possa interessar para a interpretação das respostas dos comandos solicitados.

Ao enviar comandos para os dispositivos de campo a aplicação deve encapsular cada byte na ordem correta, formando o pacote de acordo com o estipulado na norma *HART*, caso contrário o dispositivo não responderá adequadamente ao comando. Este controle da ordem em que são encapsulados os dados precisa ser feito em cada comandos, pois algumas informações nos campos variam de acordo com o comando. Para respeitar o

estipulado pela camada de transporte o código foi escrito separando cada comando em um arquivo diferente – *Unit.* Em cada *Unit* do *Handheld* o comando é tratado, encapsulado e verificada a resposta de acordo com o descrito na DDL e como esperado pela camada de transporte.

**Camada de Aplicação**

Por fim temos a implementação da última camada do modelo OSI, a camada de aplicação. Como explicado no capítulo anterior, o *HART* é um protocolo orientado a comandos, o que significa que qualquer informação necessária deve ser solicitada ao dispositivo de campo via codificação de comandos. Cabe à aplicação enviar o comando e os parâmetros adequados para cada informação solicitada, para haver entendimento por parte dos dispositivos com os quais está se comunicando.

Para certificação que cada comando está sendo enviado corretamente, assim como que a sua resposta está sendo recebida e processada adequadamente, cada comando possui a sua própria função, com métodos próprios para verificação da resposta recebida. Assim cada comando insere os parâmetros adequados na requisição e verifica os campos pertinentes buscando a resposta desejada.

Algumas verificações devem ser feitas para qualquer comando. Por exemplo, deve ser verificada na resposta de todo comando se o comando respondido é o mesmo que o solicitado. Conforme a norma, quando o dispositivo executa corretamente um comando, responde com o número deste comando no campo *Command* no caso de comandos até o 255, e responde com o número 31 no *Command* e o número do comando estendido nos dois primeiros bytes de *Data* caso o comando seja maior do que 255.

Outras verificações de resposta variam de acordo com o comando solicitado, pois dependendo do comando em questão, caso haja problema na execução do comando por parte do dispositivo, este sinalizará pelos parâmetros específicos de cada comando na resposta. A norma apresenta tabelas com os códigos de erro de cada comando, assim como diz em quais campos estes sinais podem ser verificados. Tanto as verificações comuns para todos os comandos quanto as específicas devem ser feitas para garantir a correção e dar segurança para a comunicação.

## 5.4 Configurações da Comunicação Serial

Ao utilizar o aplicativo do *Handheld* a primeira janela com a qual o usuário se depara é a caixa – *GroupBox*– ‘Configuração da comunicação serial’, mostrado na Figura 15. Os botões que aqui se encontram são os únicos que utilizam as funções do arquivo *Serial.cpp.* Todas as outras iterações com usuário utilizarão as funções em arquivos de comandos específicos.

Estes botões são diferenciados também por terem uma ordem definida que devem ser utilizados, a fim de conseguir a comunicação com o nó da rede. O primeiro botão a ser utilizado deve ser o ‘Enumerar’, que lista na caixa preta ao lado dos botões – *ListBox* – todas as portas COM encontradas no Windows. Em geral, a porta que corresponde ao dispositivo de campo é a última da lista. Para selecionar uma porta neste *ListBox* basta clicar em cima da porta desejada.


**Figura 15. Configurações da comunicação serial.**

Em seguida utiliza-se o botão 'Abrir Porta', e a partir deste momento as respostas do programa serão mostradas na caixa de diálogo em cima deste *GroupBox*, um *Memo*, mostrado na Figura 16. Obtendo sucesso, esta função mostrará qual porta foi selecionada e em seguida mostrará 'Porta aberta com sucesso'. O próximo botão a ser utilizado é o 'Conectar', e caso esta operação seja bem sucedida, na caixa de diálogo aparecerá algumas informações do dispositivo conectado: Unique ID, Device ID, TAG curto e TAG longo, nesta ordem.

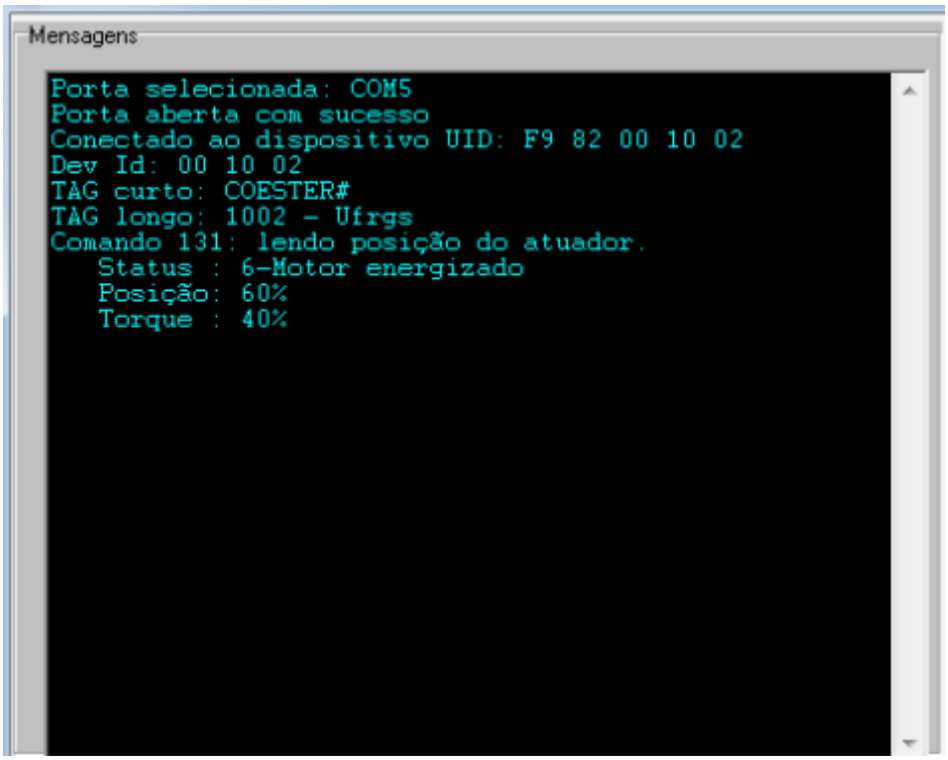



**Figura 16. Caixa de diálogo.**

Os botões ‘Conectar’ e ‘Abrir porta’ estão separados para facilitar a visualização de possíveis erros. Caso seja possível abrir a porta, mas não seja possível se conectar ao dispositivo, provavelmente o problema é com a conexão física ou com o dispositivo. Caso não seja possível abrir a porta, o problema provavelmente está relacionado com o funcionamento do sistema operacional, que por algum motivo não está identificando ou abrindo corretamente a porta COMM ligada ao dispositivo. Salienta-se que o botão ‘Conectar’ não está relacionado com a conexão física da rede, e sim com enviar comandos a fim de captar a identificação do dispositivo.

## 5.5 Configurações de Rede

Uma importante funcionalidade de um *Handheld* para redes *WirelessHART* é capacitar os dispositivos para que entre em uma determinada rede modificando os parâmetros *Join Key* e *Network ID.* Esta funcionalidade faz parte do comissionamento do dispositivo, motivo pelo qual ela é tão importante. Todo dispositivo *WirelessHART* compatível que possa ter sua porta de manutenção ligada ao computador via conexão USB pode ser configurado por estes botões de *Network*, sendo estas funções, portanto, funções básicas de software para porta de manutenção. A caixa de configurações de rede é mostrada na Figura 17, a seguir.

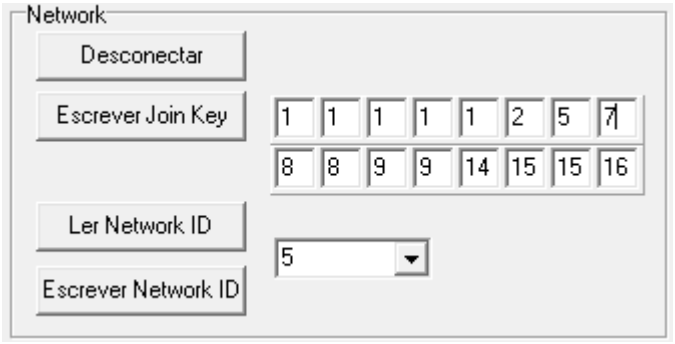



**Figura 17. Configurações de rede.**

Com o botão ‘Ler *Network ID*’ o usuário pode verificar se o dispositivo está na rede correta. É útil caso existam mais de uma rede ativa, assim podemos verificar em qual das redes encontra-se este dispositivo. O botão ‘Escrever *Network ID*’ utiliza o valor do *ComboBox* ao lado deste botão enviando um comando para o dispositivo informando que este deve ser seu novo *Network ID*. Ao clicar em ‘Ler *Network ID*’, este *ComboBox* recebe automaticamente o valor de *Network ID* lido. Para escrever um novo valor basta clicar com o *mouse* em cima do valor lido e escrever o novo valor ou ainda selecionar um novo valor na seta para baixo do *ComboBox*. Clicando na seta para baixo abrem algumas opções de números como sugestão que podem ser selecionados com um click.

O botão *Desconectar* é necessário no caso em que o dispositivo já esteja em uma rede e deva ser transferido para uma outra. Caso o dispositivo já esteja em um rede ele precisa ser desconectado para que suas *Join Key* ou *Network ID* sejam modificadas, caso contrário o dispositivo retorna mensagem padrão de erro e não é possível configurá-lo. Após um curto espaço de tempo (alguns segundos) depois de executado o comando para ser desconectado da rede, o dispositivo automaticamente tenta conectar-se novamente com os dados de rede que estiverem salvos. Caso os dados tenham sido modificados

corretamente, ele entrará na nova rede. Caso os dados não tiverem sido modificados, ele se conectará novamente na mesma rede em que estava anteriormente.

## 5.6 Potência RF

Potência RF é a potência com a qual o dispositivo enviará mensagens para seus vizinhos e para o *Gateway*. Ao clicar no botão 'Ler' o *ComboBox* é automaticamente preenchido com o valor de potência configurado atualmente no dispositivo. O valor lido também aparece na caixa de diálogo *Memo* em dBm. Para escrever um novo valor podemos selecionar algum entre as sugestões clicando na seta para baixo, ou clicar em cima do *ComboBox* e escrever um novo valor. Mas se o valor escrito não for um dos valores possíveis para o dispositivo, o novo valor será o valor mais próximo possível, mas sempre para mais próximo do zero do que o valor dado.

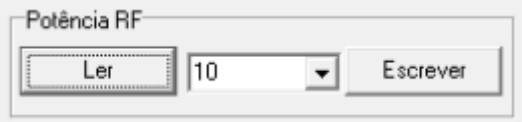



**Figura 18. Configurações da potência de saída na antena.**

A Figura 18 mostra a parte do software dedicada aos comandos para ler e escrever a potência de saída. Os valores pré-selecionados como possíveis, que aparecem como sugestão clicando na seta para baixo, estão todos na faixa de -10dBm até 20dBm.

## 5.7 Unique ID

Esta caixa apresenta o valor do *Unique ID* que é atualizado automaticamente quando a ferramenta conecta-se ao dispositivo. A *Unique ID* é um identificador único do dispositivo, sendo por este número reconhecido pelo *Gateway* e pelo gerenciador da rede. Este número é composto de 5 bytes, sendo sempre os dois primeiros, pelo padrão adotado no projeto, F9 82 em hexadecimal, como pode ser visto na Figura 19.

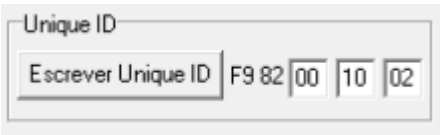



**Figura 19. Configurações de *Unique ID*.**

Além de ler o *Unique ID* do dispositivo, podemos ainda modificar este identificador, escrevendo novos valores em cima dos que ali se encontram e apertando no botão 'Escrever Unique ID'. A resposta enviada pela dispositivo é colocada no *Memo* (Diálogo de mensagens). Caso não tenha sido possível escrever novos valores, como no caso do novo valor escrito pelo usuário conter letras que nada signifiquem na representaçao hexadecimal, uma mensagem de erro aparecerá na *Memo* e os valores originais aparecerão novamente nos campos correspondentes.

Normalmente a *Unique ID* do dispositivo tem seu valor configurado em fábrica, não é comum a modificação desse valor, visto que este valor é um identificador único e se houver mais de um dispositivo na planta com o mesmo valor o *Gateway* não distinguirá os

dispositivos. Portanto, podemos modificar a *Unique ID* de qualquer dispositivo *WirelessHART* com o *Handheld*, mas devemos ter cuidado ao fazer isto. O custo de comissionar os dispositivos sem a devida atenção seria ter sérios problemas ao tentar conectar os dispositivos na rede. Na prática foi verificado que os dois dispositivos entram na rede, mas entrando e saindo, sem que nenhum dos dois fique operacional, visto que o *Gateway* pode enviar para um o comando que seria para o outro.

## 5.8 Monitoramento de Rede

Este Groupbox, mostrado na Figura 20, é útil para monitorar qualquer rede WH, visto que em qualquer rede podemos colher dados e criar estatísticas com os dados que aqui podem ser obtidos.


**Figura 20. Monitoramento de rede.**

Os três primeiros botões, quando acionados, criam uma tabela no espaço em branco ao lado do *Memo,* no *StringGrid* . O quarto botão, 'Ler Burst Mode' coloca as informação de *Burst* no próprio *Memo.* As informações obtidas com este comando são basicamente o valor Burst definido e se este está ativo. Ter Burst mode ativo significa que quando a variável primária deste dispositivo atingir o valor denominado valor de Burst o dispositivo automaticamente enviará um comando para o *Gateway* alertando de que o valor de Burst foi atingido. É como se fosse um Warning que pode ser configurado para o sensor alertar a planta de algo que não deveria estar acontecendo.

O botão 'Ler Links', conforme ilustrado na Figura 21, lista todos os links em que este dispositivo pode se comunicar. Mostra com qual sensor ele está se comunicando em cada link (Nickname), a qual *superframe* este link pertence (*Superframe* ID), a qual o *time slot* este link está associado (Time Slot), a frequência em que este link está se comunicando no momento (Ch Offset), se este link é para receber ou enviar dados (Link Opt) e ainda a qual tipo de atividade este link está relacionado (Link Type).

| Link Index | Superframe ID | Time Slot | Ch offset | Nickname | Link Opt | Link Type |
|---|---|---|---|---|---|---|
| 0 | 1 | 0A | 9 | FFFF | 2-RX | 2-Broad |
| 1 | 0 | 325 | 3 | 01 | 1-TX | 0-Normal |
| 2 | 0 | 01 | 0 | FFFF | 3-RX/TX | 1-Discov |
| 3 | 1 | 1E | 1 | F980 | 1-TX | 3-Join |
| 4 | 0 | 25 | 9 | 01 | 1-TX | 0-Normal |
| 5 | 1 | 28 | 6 | FFFF | 2-RX | 2-Broad |
| 6 | 4 | 2F | 13 | FFFF | 1-TX | 2-Broad |
| 7 | 0 | 125 | 11 | 01 | 1-TX | 0-Normal |
| 8 | 0 | 1D7 | 10 | F980 | 2-RX | 3-Join |
| 9 | 0 | 225 | 11 | 01 | 1-TX | 0-Normal |
| 10 | 0 | 3D7 | 11 | 02 | 2-RX | 0-Normal |

**Figura 21. *StringGrid* com tabela de resposta ao botão 'Ler Links'.**

O botão 'Ler *Superframe*s', como é mostrado na Figura 22, lista todos os *superframe*s que o dispositivo consegue perceber na rede. Lista o nome do *superframe* (*Superframe* ID), a quantidade de slots do *superframe* (# of Slots) e ainda mostra o tipo de *superframe*, se comum ou especial (Mode Flags).

| Superf Index | Superframe ID | # of Slots | Mode Flags | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 256 | 11 | | | |
| 1 | 0 | 1024 | 11 | | | |
| 2 | 4 | 128 | 11 | | | |

**Figura 22. *StringGrid* com tabela de resposta ao botão 'Ler *Superframe*s'.**

O botão 'Ler Vizinhos', como é mostrado na Figura 23, lista todos demais dispositivos vistos na rede pelo dispositivo conectado ao *Handheld*. Na tabela aparece o nome dado pelo *Gateway* para o vizinho encontrado (Nickname Vizin), se o vizinho é um *time source* da rede, ou seja, se este é referência para atualizar *clock* na rede (Flags), qual a potência média com que está escutando este vizinho (Mean RSL), a quantidade de pacotes transmitidos para este vizinho (Pcks Transm), destes pacotes enviados, quantos ainda não tiveram ack recebidos (acks perdidos) e quantos pacotes foram recebidos deste vizinho (Pcks Receb).

| Index | Nickname Vizir | Flags | Mean RSL | Pcks Transm | acks perdido | Pcks Recebi |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 177 | 119 | 1 | 97 |
| 1 | 2 | 0 | 193 | 59 | 0 | 1 |

**Figura 23. *StringGrid* com tabela de resposta ao botão 'Ler Vizinhos'.**

## 5.9 Configurações da Válvula

No caso específico do hardware desenvolvido pelo projeto da Petrobrás, que atuará na abertura e fechamento de válvulas, o *Handheld* serve também para enviar comandos específicos para estes dispositivos. O envio destes comandos específicos é previsto em norma, que guarda alguns números de comandos para serem implementados de acordo com as especificidades de cada dispositivo.

Para controlar e modificar o nível de abertura das válvulas foram implementados alguns comandos específicos, utilizados nesta caixa. A aplicação permite ao usuário facilmente interagir com o hardware e, por meio desse, ajusta a maquinário (no caso a válvula) modificando a abertura da válvula de forma segura, não tendo que agir no painel da válvula. Assim, o usuário não precisa saber utilizar o painel da válvula para configurá-la. A Figura 24 mostra a parte do software dedicada às configurações da válvula.

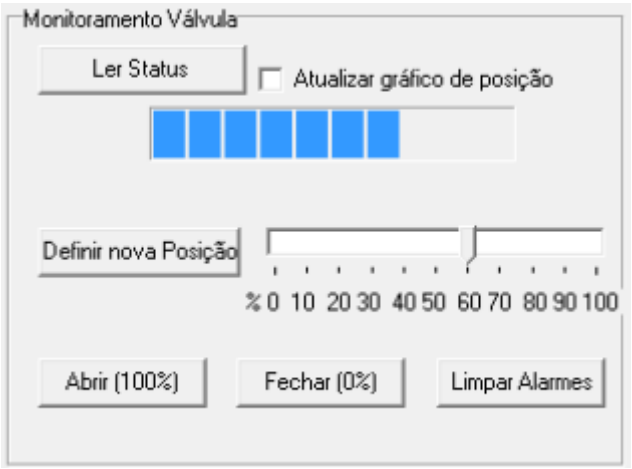



**Figura 24. Configurações da Válvula.**

Com um simples apertar de botão o usuário pode abrir ou fechar completamente a válvula, utilizando os botões 'Abrir (100%)' e 'Fechar (0%)'. A porcentagem dos botões referem-se ao nível de abertura que a válvula irá adquirir. Caso se deseje abrir ou fechar a válvula até um nível de abertura intermediário, existe uma barra de controle em que o ponteiro é ajustado com o *mouse*. Após selecionar na barra a nova posição desejada, basta clicar no botão 'Definir nova Posição'. Na Figura 24 esta barra de controle está fixada em 60%, assim ao apertar o botão 'Definir nova Posição' o nível de abertura da válvula irá ser redefinido para 60%, abrindo ou fechando a válvula se preciso.

Este *Groupbox* possui um também um visor que mostra graficamente qual a abertura atual da válvula e que pode ser atualizado constantemente (a cada 1 segundo) se marcada a opção *atualizar gráfico,* ajudando a monitorar a abertura ou fechamento da válvula ligada ao dispositivo.

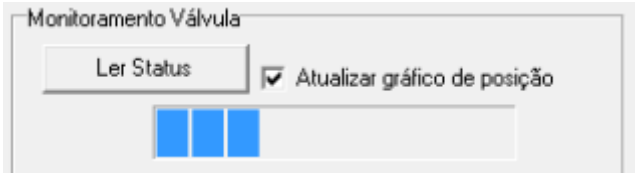



**Figura 25. Gráfico com atual nível de abertura da válvula.**

Ainda existem os botões ‘Limpar Alarmes’ e ‘Ler Status’. O botão ‘Limpar Alarmes’ limpa os alarmes internos da válvula. É necessário para que antigos alarmes da válvula não influenciem no diagnóstico atual da válvula. O botão ‘Ler Status’ coloca informações no *Memo* lidos da válvula. Coloca o status da válvula, a posição de abertura da válvula em porcentagem e o torque do atuador. Na Figura 26 é mostrada a caixa de mensagens com a resposta obtida pelo usuário ao pressionar o botão ‘Ler Status’.

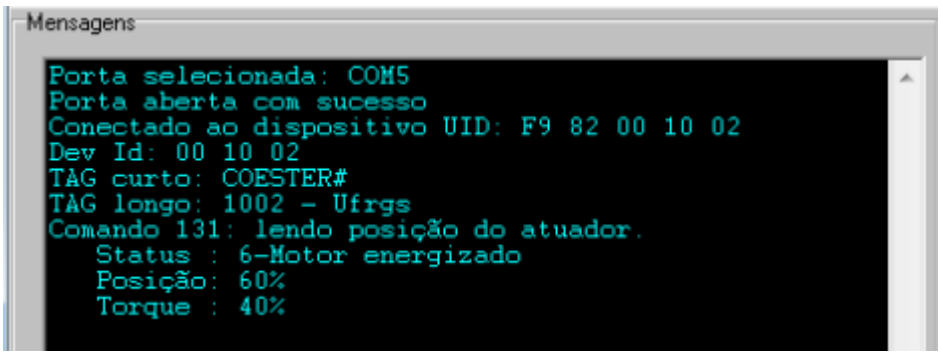



**Figura 26. Caixa de diálogo com informações de resposta do botão ‘Ler Status’.**

## 2.4 5.10 Modelo dos comandos

Para exemplificar a implementação dos comandos será mostrado o funcionamento de uma função ao ser solicitada via *form* de iteração com o usuário. Tomando como exemplo o botão ‘Escrever Unique ID’, será descrito o procedimento dentro do software até que chegue a resposta na caixa de diálogo do próprio *form.* A Figura 27 mostra o botão ‘Escrever Unique ID’, cujos passos para a realização dos comandos serão demonstrados.


**Figura 27. Botão ‘Escrever *Unique ID’*.**

Ao clicar no botão ‘Escrever Unique ID’, a função ‘Write_Unique_id’ é chamada pelo arquivo *Unit1.cpp.* Esta função faz parte do arquivo *cmd124.cpp,* e os parâmetros passados para ela são os campos preenchidos pelo usuário para ser o novo valor de

*Unique ID*. A seguir, na Figura 28, é apresentado o código da função 'Write_Unique_id' que será analisada.

```
//---------------------------------------------------------------------------
bool Write_Unique_id (const char *primeira,const char *segunda,const char *terceira )
{

  int prim, seg, terc;
  prim = hextoint (primeira);
  seg = hextoint (segunda);
  terc = hextoint (terceira);


  unsigned short    index_data;
  unsigned char     frame_len;
  unsigned short    response_data_index = RSP_DATA_INDEX_CMD;

  MainForm->Memo_Comm->Lines->Add("Comando 124: escrevendo unique ID");
  // Initialize structures

   index_data = MainForm->buildHeader(WRITE_UNIQUE_ID);
     //2 PRIMEIROS BYTES SEMPRE SÃO F9 E 82 (hexa)
    MainForm->m_ucTxBuffer[index_data] = 249; //=f9h
    index_data++;
    MainForm->m_ucTxBuffer[index_data] = 130; //=82h
    index_data++;
    MainForm->m_ucTxBuffer[index_data] = prim;
    index_data++;
    MainForm->m_ucTxBuffer[index_data] = seg;
    index_data++;
    MainForm->m_ucTxBuffer[index_data] = terc;
    index_data++;
    frame_len = (unsigned char)index_data + 1; // count FCS

    if (frame_len > FRAME_MAX_SIZE)
        return FALSE;

    if (!MainForm->buildFCS((unsigned char)index_data))
            return FALSE; // Error on processing FCS
    // End of request data

    if(MainForm->execHARTCmd(frame_len)){
        unsigned char *p;
        p = MainForm->m_ucRxBuffer+response_data_index-7;
        if (p[5] == 0 && p[3] == 124 )    // p[5] = response code -> 0 = success; p[3] =
        {
              MainForm->Memo_Comm->Lines->Add("   done " );
        }
        else {
            MainForm->Memo_Comm->Lines->Add("   Response code = " + IntToStr(p[5]));
            return FALSE;
        }
        if (p[0] == 8)
            MainForm->Memo_Comm->Lines->Add(" Warning: Set to nearest possible value.");
        return TRUE;
    }
    return FALSE;
}
```

**Figura 28. Código da função 'Write_*Unique_id*'.**

Primeiramente os três valores de parâmetro passados são convertidos de hexadecimal para números decimais inteiros pela função ‘hextoint’, função implementada neste mesmo arquivo *cmd124.cpp*. Com os valores em decimal podemos avisar ao usuário que o comando começou a ser executado, enviando a seguinte mensagem para a caixa de diálgos: “Comando 124: escrevendo unique ID”.

Caso não tenha sido possível converter os valores escritos pelo usuário para números decimais uma mensagem de erro será enviada ao usuário ao invés desta última. A primeira ação para formar um *frame* de comunicação *WirelessHART* é construir o *header*, o cabeçalho com bytes correspondentes ao endereço do dispositivo, ao preâmbulo, e até mesmo ao número do comando que será solicitado. Para tal é chamada a função ‘BuilHeader’, da *Unit1*, que tem como parâmetro o número do comando a ser solicitado ao hardware, sendo este neste caso o número 124, mas que é passado através de um *define*, uma variável estática, cujo valor não pode ser modificado no decorrer da execução. Este *define* do comando em questão é chamado de *‘WRITE_UNIQUE_ID’*.

Em seguida são incluídos os cinco bytes de parâmetros do comando 124 no buffer de saída, e a cada espaço ocupado no buffer é atualizado o ponteiro que indica qual o próximo espaço no buffer a ser preenchido. Para verificar quantos bytes são esperados de parâmetro, assim como para saber quais serão os bytes de resposta, deve ser observada a especificação do comando [4]. A especificação do comando 124 é mostrada a seguir, na Figura 29.

### Comando 124 – Write Unique ID

Escreve Unique ID do dispositivo. Os 5 bytes do UID é formado pelo device ID e pelo Extended Device Type (atribuído pela HART Foundation).

Dados da Requisição

| Byte | Formato | Descrição |
|---|---|---|
| 0 - 1 | Enum | Extended Device Type (MSB-LSB) (Table 1) |
| 2 - 5 | Unsigned-24 | Device ID (MSB..LSB) |

Dados da Resposta

| Byte | Formato | Descrição |
|---|---|---|
| 0 - 1 | Enum | Extended Device Type (Table 1) |
| 2 - 5 | Unsigned-24 | Device ID |

**Figura 29. Especificação do comando 124.**

Os parâmetros do comando inseridos no buffer serão passados como os dados do campo *Data* do *frame* de comunicação. Após inserir todos os parâmetros é feita uma verificação padrão para assegurar que o tamanho do *frame* ainda está dentro do tamanho limite estipulado por norma.

Para montar o *frame* por inteiro ainda é necessário o cálculo do *CRC*, necessário para controle de possíveis erros de envio. Para este cálculo é chama outra função da *Unit*, a

*BuilFCS*. A seguir o *frame* pode ser enviado para o hardware via outra função da *Unit1*, a '*ExecHARTCmd'*, que tem como parâmetro apenas o tamanho do *frame* a ser enviado.

Dentro da função '*ExecHARTCmd'*, após enviar o *frame* é chamada a função que recebe a resposta do dispositivo, a '*waitForResponse'*. Esta função funciona como uma máquina de estados, o software reconhece o preâmbulo que é descartado, reconhece o delimitador e recebe mais sete bytes até o byte que diz quantos bytes ainda terá a mensagem. A partir de então são esperados a quantidade de bytes declarada. O código dessa função será mostrado na Figura 30. Após obter resposta do dispositivo, esta função retorna uma *flag* com TRUE, e a função 'Write_Unique_id' então segue sua execução para verificar a resposta do dispositivo.

A verificação neste caso consiste em buscar no *frame* de resposta o byte correspondente ao campo *Command* e ao primeiro byte do campo *Data*. Devemos verificar se o campo *Command* é igual a 124, garantindo que o dispositivo entendeu o comando solicitado, e verificar se o byte de resposta no campo *Data* é igual a zero, garantindo que a solicitação foi processada com sucesso. Para entender melhor os parâmetros que devem ser enviados e o que esperar de resposta, precisamos olhar a norma com o comando específico.

```
while((count <= Length) && (ucTimeOut))
 {
     ClearCommError(m_hComFile, &dwErrorFlags, &strComStat);       //COMSTAT (strCom
     dwLength = min(255, strComStat.cbInQue);                     //cbInQue : Especi
     ReadFile(m_hComFile, &ucBuffer[0], dwLength, &dwLength, &osRead);
    if ((count >= 7)&& (mudou == 0) )
    {
        Length = 7 + (int (m_ucRxBuffer[7]));
        mudou++;
        ucTimeOut = (unsigned char)((Length * 9.16 + 300.0f) / 10.0f);
    }
     if (dwLength)
     {
         // Data was received

         for(e = 0; e < dwLength; e++)
         {
             switch(ucRcvStatus)
             {
                 case WAIT_PREAMBLE:
                     if(ucBuffer[e] == 0xff)
                     { // Preamble byte
                         ucRcvStatus = WAIT_DELIMITER;
                     }
                 break;
                 case WAIT_DELIMITER:
                     if(ucBuffer[e] == 0xff)
                     { // Preamble byte do nothing
                     }
                     else
                     {

                         if((ucBuffer[e] &0x3f) == 0x06)
                         { //Delimiter begin recording data
                             m_ucRxBuffer[0] = ucBuffer[e];
                             m_ucRcvLen++;
                             ucRcvStatus = READING_DATA;
                             //Memo_Comm->Lines->Add("buf" + IntToStr(m_ucRxBuffer[
                         }
                         else
                         { // Garbidge wait for next preamble
                             ucRcvStatus = WAIT_PREAMBLE;
                         }
                     }
                 break;
                 case READING_DATA:
                      m_ucRxBuffer[m_ucRcvLen++] = ucBuffer[e];
                      //Memo_Comm->Lines->Add("buf" + IntToStr(m_ucRxBuffer[m_ucRcvl
                      count ++;
                 break;
             }
         }
     } // if dwLength
     ucTimeOut--;
     Sleep(50); // sleep 10 ms
 } //while
```

**Figura 30. Código da máquina de estados de recepção de dados.**

# 6 CONCLUSÕES E DESAFIOS FUTUROS

O objetivo do projeto foi alcançado, visto que o *Handheld* desenvolvido está funcionando perfeitamente, podendo ser utilizado não só no laboratório mas também em fábrica para comissionamento dos dispositivos. O *Handheld* que se liga aos dispositivos via cabo USB é versátil, podendo se ligar a qualquer tipo de porta de manutenção de dispositivos *WirelessHART* dês de que se obtenha um conversor USB da referida entrada.

Da forma em que está estruturado, o software pode ser ampliado facilmente. Seguindo a organização dos comandos já implementados é possível implementar qualquer outro comando seguindo a ordem de montagem, envio e recepção de *frames*. Assim, o *Handheld* pode ser modificado e ter suas funcionalidades ampliadas a qualquer momento, de acordo com as necessidades dos dispositivos adquiridos.

A maior dificuldade na implementação do *Handheld* foi a temporização do recebimento das mensagens. Inicialmente o software recebia bytes até detectar um certo tempo sem receber mais bytes, que simbolizava que o *frame* havia sido inteiramente recebido. Desta forma muitos pacotes eram perdidos, e muitos recebidos pela metade, pois qualquer atraso no envio ou recebimento causava a quebra do *frame*. Agora o software reconhece o preâmbulo que é descartado, reconhece o delimitador e recebe mais sete bytes até o byte que diz quantos bytes ainda terá a mensagem. A partir de então são esperados a quantidade de bytes declarada. A comunicação ficou mais segura desta forma, pois espera até que todos os bytes sejam recebidos e sabe exatamente onde começa e abaca cada *frame.*

Mais um problema de temporização na recepção seria que ao esperar os bytes é dado um *sleep* entre o recebimento de dois bytes. Na maior parte dos computadores o software está funcionando corretamente, mas ao instalá-lo em computadores com velocidade muito diferente de processamento do computador em que foi desenvolvido, como ao instalá-lo em um computador antigo, está havendo pequenas perdas de pacote. A solução seria retirar o *sleep* como forma de temporização e utilizar funções específicas da API do Windows para sinalização que um byte foi recebido. Para esta correção estão sendo estudadas soluções como inserir um '*WaitCommEvent*' imediatamente antes do '*ReadFile*', configurando o máximo tempo de espera com um '*SetCommTimeouts*' e inserindo o evento a ser esperado com '*SetCommMask*'.

# REFERÊNCIAS

[1] Caro, D. : Wireless Networks for Industrial Automation. [S.l.]: ISA Press, 2004.

[2] A. Willig, K. Matheus, A. Wolisz. : Wireless Technology In Industrial Networks.

Proceedings of the IEEE, Vol. 93, No. 6, June 2005.

[3] *HART* Communication. Disponível online em:

http://www.*HART*comm2.org/index.html.

[4] *HART* (Ed.) *HART* Communication Foundation, A Technical Overview, HCF_LIT-20 Revision 3.0, 2007.

[5] K . Khakpour, M.H. Shenassa, "Industrial Control using Wireless Sensor Networks". Information and Communication Technologies: From Theory to Applications, ICTTA, April 2008.

[6] *HART* (Ed.) *HART* Communication Foundation, Network Management Specification, HCF_SPEC-085 Revision 1.1, 2008.

[7] Springer (Ed). *WirelessHART* : Real-time mesh network for industrial automation. (S.1.), 2010.

[8] Muller, Ivan; PEREIRA, Carlos E.; NETTO, João C.; FABRIS, Eric C.; ALGAYER, Rodrigo : Development of *WirelessHART* Compatible Field Devices. 2010.

[9] Disponível online em: http://www2.emersonprocess.com/en-US/brands/micromotion/configuration-peripherals-tools/475-Communicator/Pages/index.aspx, em 03/12/2011

[10] Disponível online em: http://www.bluetooth.com/Pages/Bluetooth-Home.aspx;

http://www.wi-fi.org/; http://www.zigbee.org/; http://www.wina.org/pages/default.aspx , em 03/12/2011

[11] Muller, Ivan : "Co-Processamento para Descentralização do Gerenciamento de Redes sem Fio Industriais.", 2010

# ANEXO <ARTIGO DE TG1>

Este anexo é o artigo de proposta de trabalho de graduação, entregue no primeiro semestre deste ano como Trabalho de Graduação 1 (TG1):

## Desenvolvimento de um *Handheld* para rede de dispositivos *WirelessHART*

**Carolina Puggina Lima**

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

cplima@inf.ufrgs.br

***Abstract.*** *Advances in eletronics technology have allowed the growth of automation equipment for industrial processes. The deployment of a wireless network industries reduce the cost of installation of the devices while enables the expansion of the network providing communication with devices in places of difficult access to the cables. The maintenance of this network is ideally made by a specific device, he wireless Handheld. This paper is about a development of a Handheld WirelessHART, the protocol for wireless sensor networks most accepted in industry.*

***Resumo.*** *Os avanços na área de eletrônica vêm permitindo o crescimento tecnológico dos equipamentos de automação para processos industriais. A implantação de uma rede sem fio nas indústrias, além de reduzir o custo de instalação dos dispositivos, possibilita a expansão da rede provendo comunicação com dispositivos em locais de difícil acesso para os cabos. A manutenção dessa rede idealmente é feita por um dispositivo específico, o Handheld, via comunicação sem fio. O trabalho aqui proposto consiste na implementação de um Handheld para redes WirelessHART, o protocolo para redes de sensores sem fio mais aceito no ramo industrial.*

### 1 Introdução

Os avanços na área de eletrônica vêm permitindo o crescimento tecnológico dos equipamentos de automação para processos industriais. Equipamentos cada vez mais

robustos e com maior poder computacional tornam imprescindível a comunicação entre os diversos dispositivos que compõem o cenário industrial.

Melhorias na tecnologia de sensores levaram ao surgimento de uma grande variedade de dispositivos inteligentes, dotados de microprocessadores e que fornecem uma boa visão dos processos. Quando conectados, estes dispositivos tem o poder de melhorar o desempenho operacional da planta monitorando o funcionamento de maquinários e até mesmo ajustando suas configurações, se necessário.

A implantação de uma rede sem fio nas indústrias, além de reduzir o custo de instalação dos dispositivos, possibilita a expansão da rede provendo comunicação com dispositivos em locais de difícil acesso para os cabos. Dês da década de oitenta existe grande esforço em projetar e especificar protocolos de comunicação que tornem cada vez mais seguro o controle de processos industriais. Algumas organizações tem promovido o uso de tecnologias wireless nas industrias, dentre elas Bluetooth[1], WiFi[2], Zigbee[3] e Wina[4], mas nenhuma obteve sucesso em estabelecer um padrão absoluto para as indústrias, principalmente pela falta de robustez e segurança.

O ambiente industrial é extremamente hostil. Pode apresentar ruídos eletromagnéticos de grande intensidade em alguns momentos, como no acionamento de motores elétricos. Estes ambientes também costumam apresentar temperaturas e umidades elevadas, aspectos que costumam ser prejudiciais aos componentes utilizados em sistemas computacionais e de comunicação. Neste contexto, os equipamentos designados para o uso em indústrias são especialmente construídos para atuar nestas condições adversas, assim como os protocolos de comunicação adotados também devem considerar aspectos de segurança e disponibilidade nestas condições.

À frente dos demais, a *HART* Foundation[5] lançou em 2007 a norma *HART* 7, que já contava com as seções relativas ao *WirelessHART*, sendo portanto o primeiro padrão aberto de comunicação sem fio especificamente desenvolvido para uso industrial (ORGANIZATION, 2007). Hoje, a venda de equipamentos que utilizam este protocolo está em ascensão, o que indica que o protocolo está tendo uma boa aceitação no mercado. Isto deve-se, principalmente, ao fato do protocolo ser robusto e seguro.

## 1.1 Desafio Futuro

Uma vez feita a instalação da rede Wireles*HART* na fábrica surge o interesse em técnicas de manutenção dos dispositivos desta rede. A manutenção pode ser feita por um dispositivo específico, o *Handheld*, ou por uma porta localizada no próprio dispositivo, chamada de porta de manutenção.

Um projeto visando a implementação de uma rede de sensores sem fio para medir e controlar abertura de válvulas foi encomendado pela Coester, uma empresa fornecedora da Petrobrás, para o LASCAR (Laboratório de Sistemas de Controle e Automação) da UFRGS. Os dispositivos de rede deste projeto já estão sendo feitos (tanto as placas de hardware como os firmwares) e estes encontram-se em etapa de testes. Um software para atuar como porta de manutenção também está sendo implementado, estando alguns comandos já funcionais. Este software conecta-se aos dispositivos via porta serial por meio de um conversor USB 485 para configurá-los localmente.

O desafio futuro seria desenvolver um *Handheld* wireless, um dispositivo que se comunicaria não mais por porta serial mas via comunicação sem fio, pois se um dos objetivos ao criar uma rede wireless é ampliar a rede englobando dispositivos em locais de

difícil acesso para cabos, não faz sentido a única forma de manutenção dos mesmos ser apenas por porta serial. É preciso desenvolver um *Handheld* que tenha acesso aos dispositivos da rede via comunicação *WirelessHART*.

## 2 Componentes de uma instalação *WirelessHART*

Uma rede *WirelessHART* suporta uma grande variedade de dispositivos de diferentes fabricantes. A figura abaixo ilustra os elementos básicos de uma instalação e os tipos mais comuns de equipamentos que podem estar presentes. Em alguns casos, como em um Field Device, o produto é um dispositivo físico. Em outros casos, como o Network Manager, o produto é um elemento lógico, sendo descrito abstratamente.

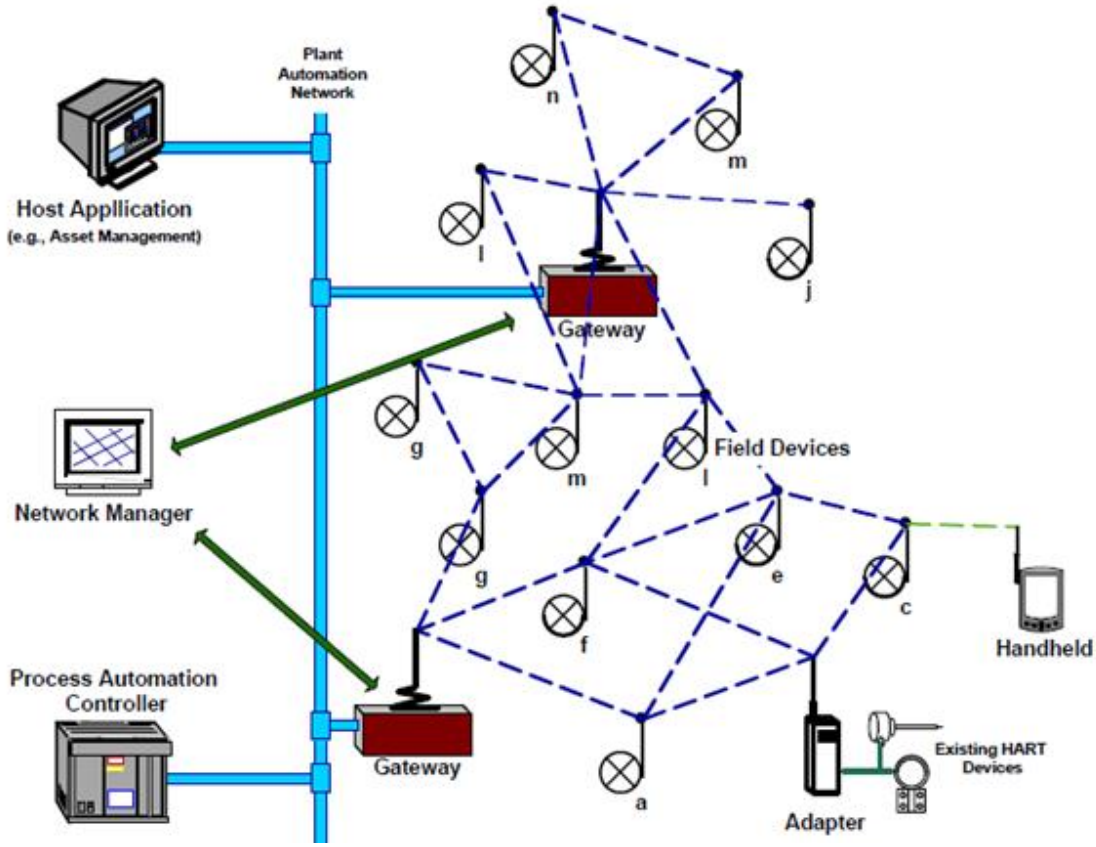



**Figura 1. Elementos de uma instalação *WirelessHART*.**

### 2.1 Dispositivos de Rede

Dispositivos de rede (*Network Devices*) são todos os dispositivos que possuem ligação na camada física da rede. Cada dispositivo de rede possui um endereço único (*Unique Adress*) para comunicar-se na rede e uma lista de vizinhos, identificados em operações de escuta da rede. Estes vizinhos podem ser identificados em qualquer slot, ou seja, há qualquer momento. Os dispositivos de rede mais comuns são os dispositivos de campo, adaptadores e *Gateway*s.

Todo dispositivo de rede deve ser capaz de rotear pacotes e de entregar pacotes em nome de outro dispositivo da rede. Uma tabela interna do dispositivo é usada para decidir o roteamento que será dado aos pacotes.

### 2.1.1 Dispositivos de Campo

Os dispositivos de campo (*Field Devices*) são os que aparecem na figura nomeados por letras de '*a*' a '*n*'. Este é o tipo mais comum de dispositivo de rede, sendo eles os dispositivo que captam informações do maquinário e do ambiente e transmitem as informações pertinentes pela rede, tornando porssivel o controle de processos. No projeto em questão, os dispositivos de campo são rádios ligados a sensores que monitoram o status e a abertura e o fechamento de válvulas.

### 2.1.2 Adaptadores

O adaptador *WirelessHART* é um dispositivo de rede que adiciona a comunicação *WirelessHART* a um dispositivo que já seja um *HART* Field Device, ou seja, já possua a comunicação *HART* wired. Ele atua como um tradutor entre *HART* e *WirelessHART*, podendo o dispositivo de campo que porte o adaptador transmitir e receber mensagens em qualquer um dos protocolos.

### 2.1.3 *Gateway*

*Gateway* é o dispositivo que provê ponto de acesso para a rede. Pode haver mais de um *Gateway* por rede ou ainda mais de uma rede com o mesmo *Gateway*, desde que cada rede tenha ao menos um *Gateway*. O *Gateway* conecta a rede *WirelessHART* à planta de automação, possibilitando a troca de dados entre redes. Os dados coletados pelo *Gateway* são passados para a planta de automação via mensagens que podem ser:

- mensagens de rotina, comunicando periodicamente dados de processos e eventos;
- mensagem de status, quando um dispositivo de campo apresentar condição de processo não recomendável ou falha de funcionamento;
- mensagens de configuração e manutenção da rede, geralmente passados por bursts.

As aplicações hosts precisam do *Gateway* para se comunicarem com os demais dispositivos de rede. O *Gateway* também provê o relógio global da rede para os seus dispositivos.

## 2.2 Gerenciador de Rede

O gerenciador de rede (*Network Manager*) é uma aplicação que gerencia a rede *WirelessHART*. Para cada rede deve haver um e apenas um gerenciador de rede. Ele é o responsável por formar a rede, controlar os joins, configurar os dispositivos de rede e monitorar a saúde da rede.

O gerenciador de rede possui uma lista de todos dispositivos de rede e garante que cada um deles terá um nickname único de 16 bits. Esta lista é usada para gerar funções de rede, como a função de roteamento e de agendar comunicação entre os dispositivos de rede. O gerenciador de rede também gerencia routing tables, para monitor a saúde da rede coletando dados de performance e diagnóstico. Todas as informações são analizadas em tempo real, permitindo a reconfiguração da rede com a mesma em funcionamento quando detectado algum problema.

Este tipo de dispositivo deve ter uma rota segura ao *Gateway* da rede, pois sem a comunicação entre eles é inviável manter a rede *WirelessHART*. Nos *Gateway*s utilizados no projeto em estudo os networks managers já estão integrados com o *Gateway*, já foram comprados como se fosse um único equipamento.

## 2.3 *Handheld*

*Handheld* representa a futura geração de instrumentos para redes wireless. É pouco utilizado atualmente, mas deve passar a ser cada vez mais presente com o aumento da utilização das redes industriais. Este dispositivo serve para manutenção dos dispositivos de campo e pode estar em um computador portátil, em um dispositivo de campo reconfigurado ou ainda em Palmtops e celulares. Ou seja, *Handheld* é um computador portátil com conexão *WirelessHART* e com uma aplicação host para a configuração de outros dispositivos da rede.

Este dispositivo nômade tem acesso aos dados da planta de automação e aos status dos processos enquanto tem o poder de gerenciar, atualizar e configurar os dispositivos de rede.

Existem quatro meios do *Handheld* comunicar-se com os dispositivos da rede:

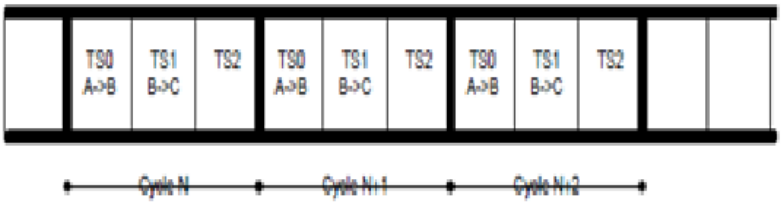
a) O *Handheld* é conectado à planta de automação via rede interna, podendo ser por uma conexão Ethernet. Assim o *Handheld* se comunica com o *Gateway* da mesma forma que faz a planta de automação. Neste caso, o *Handheld* aparece na rede como uma aplicação host.
b) O *Handheld* é conectado à porta de manutenção do dispositivo de rede. Assim o *Handheld* pode se comunicar com o dispositivo em que está ligado, mas não tem acesso a configurações de rede nem a nenhum outro dispositivo.
c) O *Handheld* é conectado na rede wireless como se fosse um dispositivo de rede, sendo visto por todos dispositivos como sendo mais um field device. Desta forma, o *Handheld* consegue comunicar-se com o *Gateway* e o gerenciador de rede e configurá-los.
d) O *Handheld* é conectado a um dispositivo de rede via a rede wireless. O *Handheld* conseguirá comunicar-se apenas com um dispositivo de cada vez. Neste caso o *Handheld* atua como sendo um dispositivo de manutenção.

# 3 Sobre o protocolo *WirelessHART*

*WirelessHART* é um protocolo de comunicação baseado em TDMA e saltos de frequência. TDMA (*Time Division Multiple Access*) é uma tecnologia de transmissão digital que permite que vários usuários utilizem o mesmo canal de comunicação pela divisão do tempo de utilização desse canal. A cada ‘fatia’ de tempo, chamadas de *time slots*, pode ser associado um *link*, uma conexão física entre dois *peers*. Assim, múltiplos nós da rede compartilhem o mesmo canal de comunicação utilizando somente uma parte da banda passante.

A coleção de todos os time slots repetindo no tempo formam um *superframe*. O número de slots de um *superframe* determina quão frequente ele se repetirá. Quando um

*superframe* é criado, no gerenciador de rede é gerado um gráfico. De acordo com esse gráfico, o gerenciador de rede vai associando links aos slots. Na figura 2 está ilustrado um *superframe* com 3 *time slots*. Cada vez que os 3 *slots* se repetem é caracterizado um ciclo. Na figura, os slots 1 e 2 possuem links, enquanto o *slot* 3 ainda está vazio.




**Fifura 2. Exemplo de um *Superframe* com 3 *time slots*.**

Os dispositivos de campo *WirelessHART* geralmente são abastecidos por uma bateria, pois mesmo que no local seja possível instalar cabeamentos, no ambiente industrial é mais seguro que o dispositivo seja alimentado por uma bateria interna do que por meio de cabos, que podem gerar curto-circuito ou pegar fogo no caso de algum maquinário soltar alguma faísca.

Desta forma, além do protocolo *WirelessHART* ser um protocolo que visa prover uma comunicação confiável, também precisa prover uma comunicação de baixo consumo. Para conseguir o baixo consumo foram pesquisadas técnicas de espalhamento espectral, originalmente desenvolvidas para uso militar. Essas técnicas permitem redução de energia por bit, redução de distorções e interferências, além de dificultar a detecção da mesma. Tipicamente nestes sistemas, o espalhamento espectral é obtido por saltos de frequência.

Assim, os *links* disponibilizados pelo *Gateway* para comunicação dos dispositivos não serão com a mesma frequência, mas a sequência de saltos de canais é conhecida pelo receptor (pelo cálculo utilizando semente do código pseudo-aleatório), então os dispositivos sabem em que frequência estabelecerão a próxima comunicação com o *Gateway*.

## 4 Etapas do desenvolvimento do *Handheld WirelessHART*

Sendo o *Handheld* um computador portátil com conexão *WirelessHART* e com uma aplicação host para a configuração de outros dispositivos da rede, como descrito na sessão *2.3*, o projeto consiste em um dispositivo de hardware funcionando como um rádio de comunicação ligado a um programa de computador via conversor USB 485, que conecta a porta USB do computador na porta serial do rádio, como ilustrado na figura a seguir.

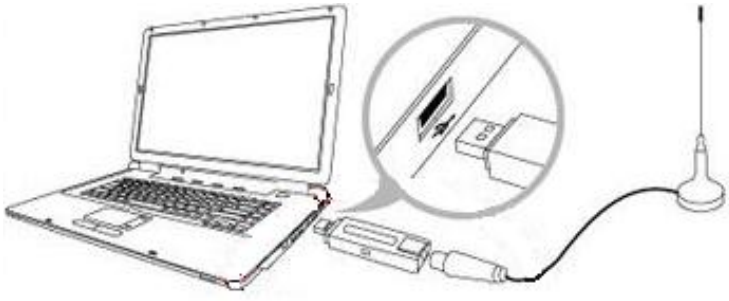


**Figura 3. Conversor USB conectando o radio ao computador.**

A construção deste rádio consiste em modificar um dos dispositivos projetados especialmente para atuar no projeto como dispositivo de campo, e a construção do software na modificação do software que está em desenvolvimento para atuar como porta de manutenção. Talvez sejam necessárias modificações também na camada de aplicação, mas para tal contamos com uma API disponibilizada pela própria *HART* Foundation.

O software que atuará como aplicação host, localizado inicialmente em um computador mas podendo ser no futuro portado para um dispositivo menor, como um celular, está sendo desenvolvido em linguagem C++ utilizando API padrão do Windows para comunicação com portas USB. A tela do programa em desenvolvimento pode ser vista na figura a seguir:

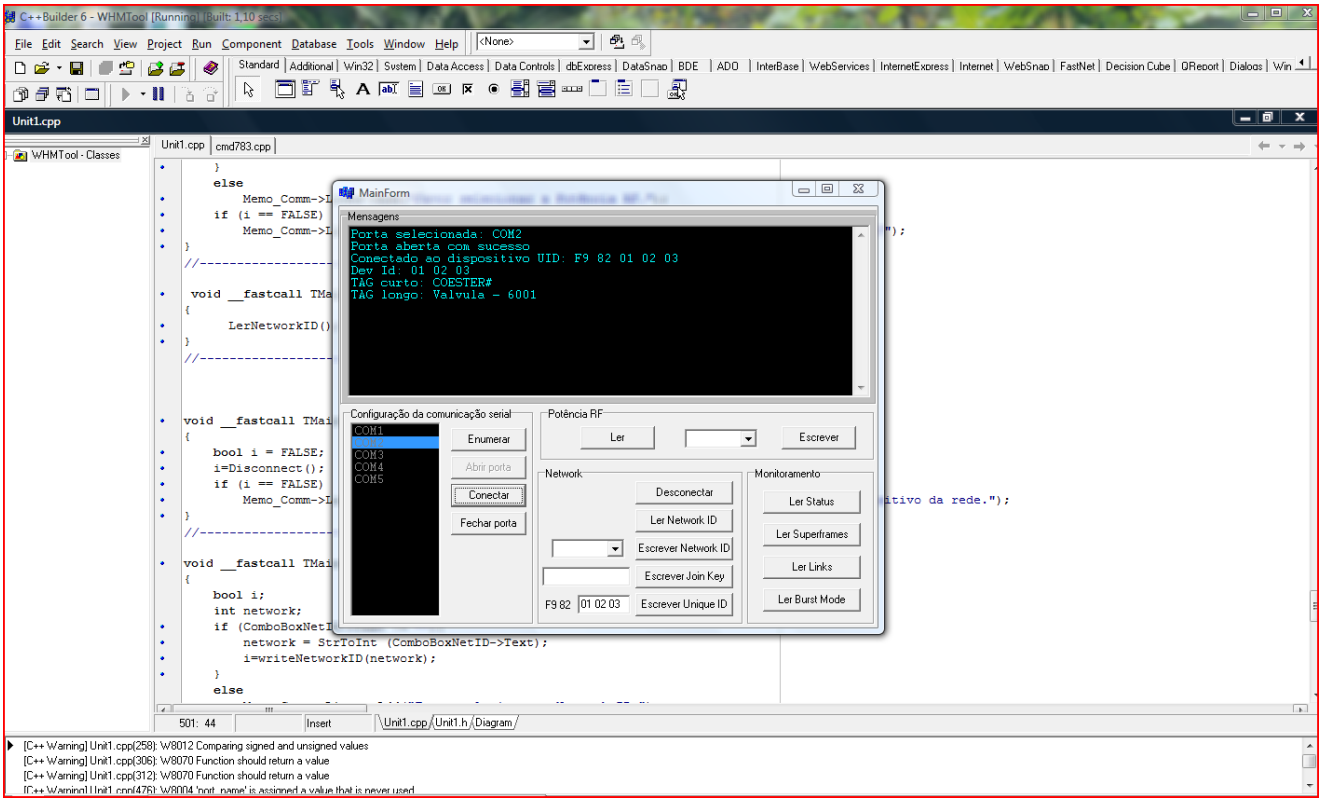



**Figura 4. Aplicação Host em fase de desenvolvimento.**

Antes da construção do *Handheld* faz-se necessário um estudo da viabilidade do projeto. Primeiramente foi verificado qual o procedimento para o *Handheld* estabelecer conexão com os dispositivos da rede. Como analisado na sessão *1.1*, para o projeto em questão é interessante desenvolver a comunicação como descrita no item *d* da sessão *2.3*, ou seja, o *Handheld* precisa conectar-se à rede sendo reconhecido como um dispositivo de manutenção. O procediento para que o *Handheld* seja reconhecido na rede desta forma está descrita na sessão *4.1* e em seguida é apresentada uma breve análise do nosso hardware atual e as modificações que seriam necessárias para a construção do *Handheld*, na sessão *4.1.1* .

Após estudo sobre o procedimento para o hardware conectar-se a rede e sobre como ele deve comportar-se na mesma, podemos analisar a API disponível e verificar se ela suporta as operações de hardware exigidas, ou seja, se ela é poderosa o suficiente para o desenvolvimento do *Handheld*. Essa análise é descrita na sessão *4.2*.

A etapa final do desenvolvimento do *Handheld* seria a implementação de um software que provesse a comunicação entre o *Handheld* e os dispositivos de rede. Este software é descrito na sessão *4.3* deste trabalho.

## 4.1 *Handheld* conectado na rede como Dispositivo de Manutenção

Uma vez que o *Handheld* esteja conectado na rede como um dispositivo de manutenção ele será capaz de coletar informações, fazer diagnósticos e calibrar os dispositivos da rede. Para suportar estas operações, a rede *WirelessHART* permite o

*Handheld* conectar-se a um dispositivo de rede por vez. Para tal, um *superframe* especial – *Handheld superframe* – deve ser utilizado, e antes disso este mesmo deve ser configurado em cada dispositivo. O procedimento para conectar-se a um dispositivo desta forma é como segue:

1. Para o *Handheld* comunicar-se com um dispositivo de rede precisa conseguir uma *session key* (chave de sessão) específica para tal. Para conseguir a chave de sessão, o *Handheld* precisa conectar-se à rede em que encontra-se o dispositivo, como se fosse um dispositivo de rede comum. Em seguida, o *Handheld* deve solicitar ao gerenciador de rede uma chave de sessão para cada dispositivo com o qual ele deseja conectar-se. O gerenciador de rede irá instalar essas chaves em cada um dos dispositivos solicitados e no *Handheld*.
2. Após obter a chave de sessão com o *Gateway*, o *Handheld* deve ser posicionado fisicamente próximo do dispositivo com o qual quer estabelecer conexão. Deve ficar próximo o suficiente para conseguir estabelecer uma conexão direta, ou seja, a antena do dispositivo deve conseguir captar os sinais do *Handheld*, assim como a antena do *Handheld* captar as mensagens do dispositivo. O *Handheld* então entra em modo de escuta procurando um sinal de *advertisement.*
3. Ao captar o sinal do dispositivo, ou seja, ao localizar o dispositivo, o *Handheld* identifica links a partir dos quais poderá estabelecer conexão com o dispositivo e envia um comando solicitando uma sessão.
4. Ao receber a solicitação de sessão, o dispositivo contata o gerenciador de rede solicitando a ativação do *Maitenance Superframe*.
5. O gerenciador de rede então ativa o *Handheld superframe* - com velocidade de transferência de dados mais alta que os *superframe*s comuns. Assim o hendheld e o dispositivo estabelecem uma conexão própria, separada da rede principal.

Nota: O *Handheld* precisa conectar-se na rede em que encontra-se o dispositivo para comunicar-se com o mesmo. Se o dispositivo não estiver conectado a nenhuma rede, o *Handheld* deve atuar também como *Gateway* e gerenciador de rede, a fim de formar sua própria rede com o dispositivo a ser configurado.[6]

### 4.1.1 Modificações necessárias para transformar um dispositivo de campo em um *Handheld*

Visto que o hardware desenvolvido como dispositivo de campo para o projeto supre as necessidades do *Handheld*, chego a conclusão que a placa do hardware não necessita mudanças; já o firmware precisa de nova programação, pois as funções são um pouco diferentes, visto que o *Handheld* deve enviar comandos configurando outro dispositivo, enquanto o dispositivo de campo deve aceitar os comandos.

Embora seja um pouco trabalhosa a modificação do firmware para adaptá-lo para o *Handheld*, é perfeitamente viável. Logo, o desenvolvimento da parte do hardware não será um empecilho para o desenvolvimento do *Handheld*.

## 4.2 Criar camada de aplicação com a API disponível para programação

A comunicação utilizando o protocolo *WirelessHART* é voltada a comandos. Cada comando possui um código e espera-se que seja seguido de um determinado parâmetro. O dispositivo ao receber o comando responde com ack ou com uma mensagem de erro. A rede *WirelessHART*, com seu gerenciador de rede e *Gateway*, já supotam a transmissão de dados da camada física até a camada de transporte. Seria necessário apenas criar a camada de aplicação específica para o dispositivo *Handheld*.

As camadas de transmissão do *WirelessHART* aparecem na figura a seguir, sendo comparadas com as camadas do modelo OSI, para melhor entendimento das equivalências com o modelo mais utilizado para transporte de dados atualmente.

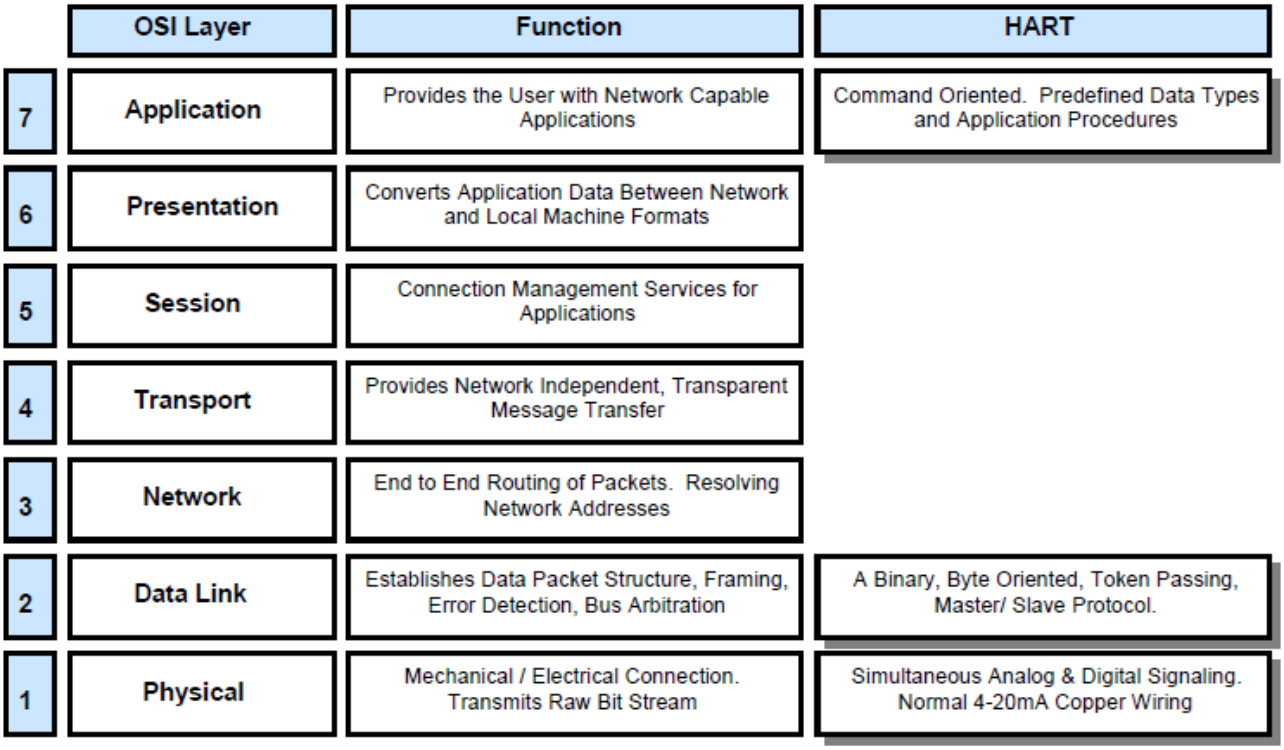



**Figura 5. Arquitetura do protocolo de comunicação *HART*.**

A biblioteca disponível para desenvolvimento é a API padrão disponibilizada pela própria *HART* Communication Foundation. Esta biblioteca não é vasta, possui apenas algumas funções básicas, mas já existe no projeto uma camada de aplicação para os dispositivos de campo, criada com esta mesma biblioteca.

A camada de aplicação necessária para o *Handheld* não é igual à camada de aplicação dos dispositivos de campo, visto que a do *Handheld* deverá suportar novos comandos e em alguns outros comandos dar tratamento diferente do que o dado nos dispositivos de campo. Porém, analisando as funcionalidades necessárias de biblioteca das duas camadas de aplicação, percebe-se que as funções necessárias de biblioteca nas duas camadas são parecidas. Logo, é possível construir uma camada de aplicação para o *Handheld* com a API disponível.

### 4.3 Implementação do Software do *Handheld*

Sendo o *Handheld* um computador portátil com uma aplicação host para a configuração de outros dispositivos da rede, faz-se necessária a construção desta aplicação host. Como mencionado anteriormente, no momento eu desenvolvo um software que serve de porta de manutenção dos dispositivos de rede. O software do *Handheld* será construído com o mesmo principio do software de uma porta de manutenção, sendo uma interface amigável ao usuário que disponibiliza todos os comandos necessários para a configuração e leitura dos dados já existentes nos dispositivos.

O software do *Handheld* wireless terá o comportamento por vezes diferente que o software da porta de manutenção. Ele terá que, por exemplo, localizar o outro dispositivo na rede, o que o dispositivo de rede não faz, e fazer uma conexão diferente do que a conexão estabelecida por porta serial. Porém, embora tenha que desenvolver mais funcionalidades no software, o software atual da porta de manutenção servirá de base para o desenvolvimento do novo.

## 5 Cronograma Previsto

| Tarefa | Prazo Estimado |
|---|---|
| Modificar firmware de um dispositivo de c | Final de Agosto. |
| Criar camada de aplicação. | Final de Setembro. |
| Finalização do software para atuar como aplic | Final de Outubro. |
| Finalização do texto para entregar com T | Final de Novembro |

## Referências (acessados em 14/06/2011)

[1] http://www.bluetooth.com/Pages/Bluetooth-Home.aspx

[2] http://www.wi-fi.org/

[3] http://www.zigbee.org/

[4] http://www.wina.org/pages/default.aspx

[5] http://www.*HART*comm.org/

[6] *HART* Communication Protocol – Block Data Transfer Specification.