4
Arquitetura Adotada

Neste trabalho foi desenvolvido um sistema para a inspeção de dutos de óleo, gás e outros fluidos. Este sistema está sendo usado em inspeções que utilizam como ferramenta de inspeção um inspetor externo de dutos - um robô que percorre o exterior de dutos, abraçando-os (Figura 1.4). O sistema permite o armazenamento e a análise dos dados obtidos pela ferramenta de inspeção, além de auxiliar no gerenciamento da inspeção (por exemplo: permite saber quais trechos do duto foram inspecionadas e quais ainda não foram; agendar os trechos do duto que devem ser inspecionados; gerar um relatório com todos os defeitos encontrados no duto durante a inspeção; determinar a gravidade de um defeito; etc.).

De forma geral, a arquitetura do sistema proposto é composta por dois elementos: um software embarcado, cujo objetivo é controlar a ferramenta de inspeção em todos os sentidos (aquisição de dados, velocidade, condições para bom funcionamento, etc.) e um software supervisor, que executa numa estação na qual o operador pode acompanhar o status da aquisição de dados e analisar os dados colhidos. A comunicação entre a ferramenta e a estação é sem fio, realizada através da tecnologia *bluetooth* com o uso de adaptadores USB. (Figura 4.1)

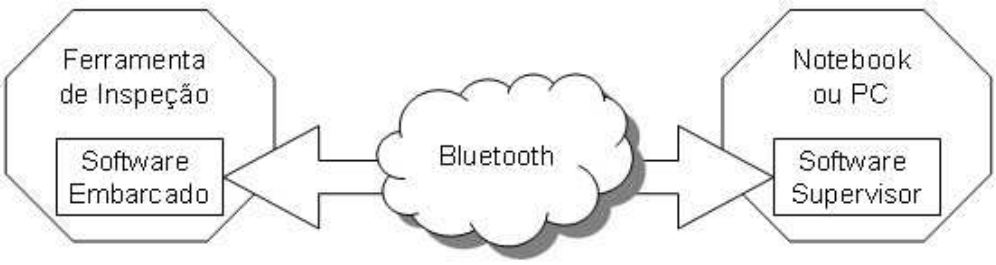

Figura 4.1: Arquitetura geral do sistema

4.1
O Software Embarcado

O software embarcado é responsável por controlar a aquisição de dados dos sensores que coletam informações do duto e monitorar o funcionamento dos mesmos. Ele é capaz de: determinar se os sensores estão funcionando corretamente; informar ao software da estação quando um sensor está com problemas; e ler os dados dos sensores regularmente enviando-os ao software supervisor.

O software embarcado possui serviços de cunho geral de forma a ser reutilizado em diferentes sistemas eventualmente desenvolvidos em linguagens de programação distintas. Por isso, foi preciso que a arquitetura utilizada e os protocolos de comunicação fossem muito bem especificados e de uso geral.

Os requisitos são os de um sistema de tempo real rígido, pois o atraso no envio ou no recebimento dos dados coletados, ou a demora no tratamento de um sensor defeituoso, acarretará perda de informações que podem fazer com que um sinal importante não seja corretamente enviado e/ou tratado a tempo.

Como o software embarcado tem um comportamento ativo, ele foi modelado como um sistema multi-agentes, onde há quatro tipos de agentes com papéis distintos: o **Agente Odométrico**, o **Agente de Sensor**, o **Agente Comunicador** e o **Agente Guardião** (ou Watchdog). A Figura 4.2 apresenta a arquitetura do software embarcado. O software embarcado possui três camadas: Camada de Comunicação, Camada de Controle do Sistema e Camada de Controle dos Sensores. A Camada de Comunicação é responsável pela comunicação entre o software embarcado e o software supervisor. Como o próprio nome diz, a Camada de Controle do Sistema gerencia o funcionamento das outras camadas e do hardware. A Camada de Controle dos Sensores é responsável pelo controle e leitura de dados dos sensores. Estes agentes serão detalhados no Capítulo 5, que fala sobre a implementação do sistema.

O software embarcado foi desenvolvido em C.

4.2
O Software Supervisor

O software supervisor é utilizado na operação da ferramenta de inspeção. É importante deixar claro que, enquanto o ambiente de execução do software embarcado é a eletrônica da ferramenta de inspeção, o ambiente de execução do software supervisor é um computador pessoal (como um notebook).

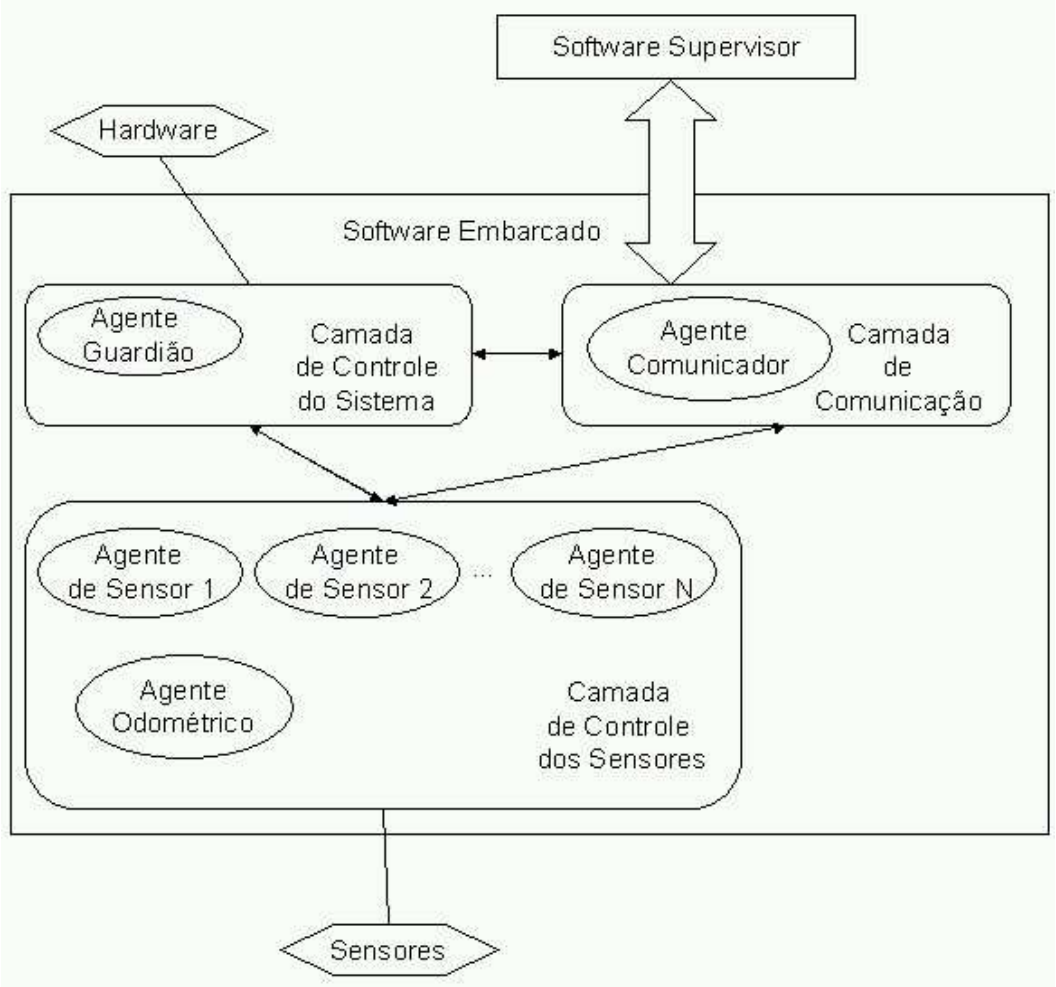

Figura 4.2: Arquitetura do software embarcado

Os dados coletados pelo software embarcado são enviados para o software supervisor, que é responsável por processá-los e armazená-los, permitindo que o operador acompanhe a análise em tempo real, e/ou realize a análise offline de dados obtidos previamente.

A fim de armazenar informações oriundas da análise dos dados (tais como localização de pontos de interesse) o software supervisor faz uso de uma base de dados com os dados da linha inspecionada. Este banco foi desenvolvido utilizando-se o SQLite[33], devido ao fato de ser um banco leve e não precisar de software ou bibliotecas adicionais. Assim, ao invés de ser um software a parte, o Sistema Gerenciador de Banco de Dados é parte integrante do software supervisor.

O software supervisor foi modelado como um sistema multi-agentes. São três os agentes que compõem o sistema: **Agente Comunicador**, **Agente Interpretador** e **Agente Documentador**. Além dos agentes existem módulos

acessórios que fornecem serviços de alto nível para os agentes. A Figura 4.3 apresenta a arquitetura do software supervisor. O software supervisor possui quatro camadas: Camada de Comunicação, Camada de Visualização, Camada de Análise de Dados e Camada de Persistência. A Camada de Comunicação é responsável pela comunicação entre o software supervisor e o software embarcado. A Camada de Visualização é responsável pela interface gráfica do sistema. A Camada de Análise de Dados é responsável pela análise dos dados lidos pelos sensores da ferramenta de inspeção (estes dados são enviados pelo software embarcado). A Camada de Persistência é responsável por gerenciar o banco de dados que armazena os dados da inspeção. Esse assunto será melhor discutido no Capítulo 5, que fala sobre a implementação do sistema.

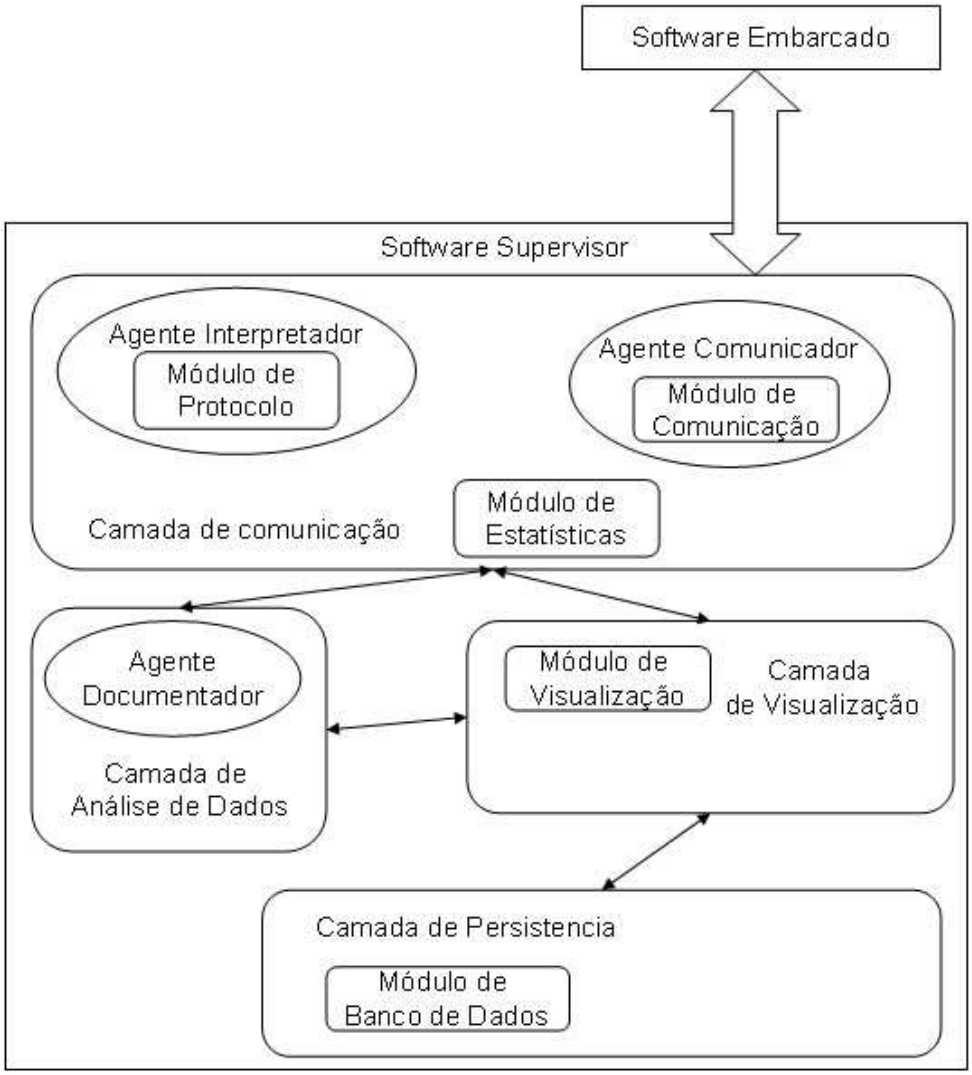

Figura 4.3: Arquitetura do software supervisor

O software supervisor foi desenvolvido em C++, para que fosse possível

reutilizar componentes de software de um sistema de análise de dados existente. Este outro sistema foi desenvolvido pela mesma equipe que desenvolveu o software supervisor. O sistema faz uso de algumas bibliotecas como o SQLite (utilizado na camada de persistência) e o Qt e o log4cxx (utilizados em todo o software).

4.3 A Comunicação

A comunicação entre o software embarcado e o software supervisor foi desenvolvida como um framework[52], onde cada parte da comunicação é tratada em um módulo específico. Estes módulos se comunicam entre si e com o resto do sistema através de interfaces bem definidas.

Módulo de Protocolo: Responsável pelo protocolo de comunicação. Ele possui as funções para codificar e decodificar as mensagens trocadas entre o software supervisor e o software embarcado.

Modulo de Comunicação: Responsável pela comunicação propriamente dita, ou seja, por enviar bytes e receber bytes. Ele cuida dos detalhes de baixo nível da comunicação.

Essa abordagem permite alterar o protocolo e/ou o meio de comunicação sem alterar o resto do sistema. Para tal é preciso criar um novo módulo (mantendo as interfaces) e substituir o antigo. Optou-se por esta abordagem devido à ausência de experiência prévia no desenvolvimento e no uso da tecnologia *bluetooth*. Seria perigoso apostar todas as fichas nesta alternativa. Da forma como foi desenvolvido (utilizando o padrão de projeto *Strategy*[53]) este módulo pode ser rapidamente adaptado a novas alternativas de comunicação, inclusive alternativas com fio.

O sistema foi desenvolvido utilizando comunicação sem fio através de *bluetooth*. Utilizou-se um adaptador USB - *Bluetooth*, comum no mercado, que cria uma série de dispositivos de entrada e saída virtuais, tais como portas paralelas e seriais, que podem ser utilizados da mesma forma que suas versões reais. O dispositivo selecionado foi o serial, ou seja, para o sistema, a comunicação é feita através de uma porta serial.

É importante ressaltar que o módulo de comunicação é genérico, podendo ser reutilizado em outros projetos.

4.4 Tecnologias Utilizadas

Tanto o software supervisor quanto o software embarcado foram desenvolvidos utilizando *Design by Contract*. Foram usadas assertivas para testar as pré e pós-condições que devem ser válidas durante a execução de funções e de pontos críticos. Além disso, foram utilizadas estruturas de dados auto-verificáveis[6], as quais possuem rotinas que auto-validam a própria estrutura e que são chamadas periodicamente a fim de minimizar o tempo decorrido desde a ocorrência de uma falha até o momento de percepção da sua ocorrência.

Mesmo sabendo que o uso de assertivas pode resultar em um aumento (as vezes significativo) do tempo de execução e que está sendo desenvolvido um sistema de tempo real, optou-se por esta abordagem devido ao fato dela facilitar a descoberta da origem dos erros, pois como o código é constantemente testado, os erros se tornam pontuais. Esta abordagem se faz necessária, pois é praticamente impossível simular uma situação real de uso do sistema, o que dificulta a repetição em ambiente de desenvolvimento de um erro ocorrido em ambiente de produção.