D'NIA: UM SISTEMA DE TEMPO REAL ORIENTADO A OBJETO

Dr. Glauco A. de P. Caurin
EESC - USP
São Carlos - SP - Brasil
glauco@precisao.sem.eesc.sc.usp.br

Dr. Sjur Vestli
Institut für Robotik - ETH
Zurique - Suiça
vestli@ifr.mavt.ethz.ch

Dr. Daniel Diez
mecos Robotics AG
Zurique - Suiça
d.diez@mecos.ch

1. Resumo

D'nia é um ambiente de programação orientado a objeto para sistemas de tempo real. Este ambiente inclui uma ferramenta para "cross-development" de sistemas dedicados, um núcleo de tempo real e um mecanismo de configuração para definir o controle de sistemas (processos).

O sistema de "cross-development" D'nia é baseado e tem a mesma filosofia do sistema Oberon, um novo ambiente de programação orientado a objeto desenvolvido no ETH Zurique, Suíça. O núcleo de tempo real é baseado em eventos e traz as dependências de tempo dos processos para um primeiro plano de importância. D'nia é utilizado como software base para controladores de robôs modulares e outros sistemas mecatrônicos e de automação.

2. Introdução

D'nia é um "cross development system" para software de tempo real que está sendo desenvolvido pela firma mecos Robotics AG e pelo Instituto de Robótica da ETH em Zurique, Suíça em conjunto com o Laboratório de Mecatrônica do Dep. de Eng. Mecânica da EESC - USP em São Carlos. O desenvolvimento do software de tempo real é feito em um "host". O computador "host" pode ser um PC, um Macintosh ou uma Workstation UNIX. O software de tempo real é desenvolvido para sistemas integrados dedicados, o sistema integrado pode, por exemplo, ser um barramento VME com múltiplas placas de processamento ou então um simples sistema microcontrolado. Uma característica especial do D'nia é o seu núcleo (kernel) de tempo real orientado a objeto. Um esforço adicional foi feito para que o D'nia se transformasse num sistema de fácil utilização. Com o D'nia acredita-se que a programação de softwares de tempo real se torne tão simples como a programação de Controladores Lógicos Programáveis (CLPs), mas mantendo o conforto e a flexibilidade da utilização de um PC ou de uma Workstation.

O desenvolvimento atual de software e hardware para PCs e Workstations é muito rápido. Os sistemas se tornam cada vez mais potentes e cada vez mais confortáveis aos usuários. As "Graphical User Interfaces (GUIs)" são rápidas,

elegantes, eficientes, fáceis de usar e intuitivas. E é por estas razões que as aplicações clássicas de tempo real migraram para o mundo das Workstations. No entanto, pode se afirmar, com respeito ao desenvolvimento de software, que os sistemas industriais (por exemplo sistemas VME) estão anos atrasados e um fator mais caro do que os produtos que podem ser encontrado no mundo dos PCs e Workstations.

Na outra extremidade do espetro computacional pode se encontrar os CLPs. Originariamente a programação de CLPs era uma programação lógica com contadores e funções de tempo. As maiores vantagens dos CLPs são a sua robustez e a simplicidade de programação. Com um tempo de aprendizado reduzido, o staff técnico pode implementar aplicações complexas de software. A capacidade das CLPs tem sido continuamente desenvolvida, apesar disto, existe uma infinidade de aplicações que não podem ser resolvidas utilizando o modelo simplificado oferecido pelos mesmos.

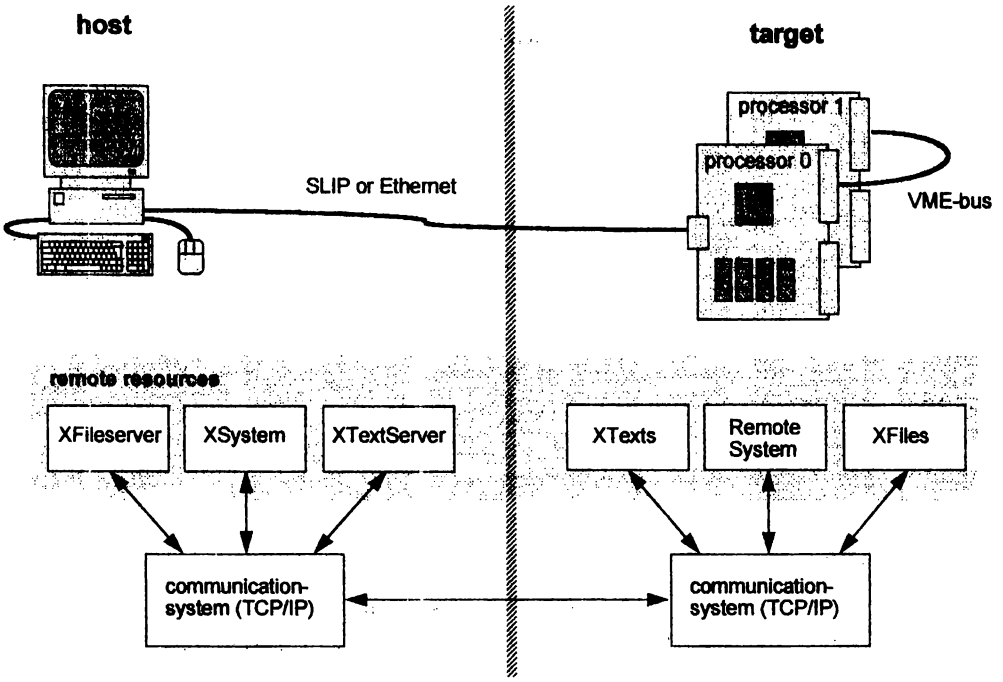

Figura 1: Sistema de comunicação (TCP/IP) entre o "host" e o "target".

Sistemas dedicados ou sistemas de controle são freqüentemente relevantes do ponto de vista de segurança e por isso mesmo devem oferecer performances de tempo real e comportar ***programação de segurança***, isto é, eles devem atender restrições de tempo do processo a ser controlado, comportar programação estruturada e controle de tipo. Portanto, o sistema de computação deve ser multi-tarefas oferecendo um tempo curto de chaveamento entre as tarefas. Além disto, ele deve permitir computação de elevada complexidade, como por exemplo o controle simultâneo dos seis graus de liberdade de um robô.

A implementação de um software complexo pede pela potência e flexibilidade dos PCs e Workstations atuais. As exigências de tempo real, por outro lado, pedem

por simplicidade, potência e confiabilidade de "funções de tempo" como as que podem ser encontradas em CLPs. Adicionalmente, é de importância vital para sistemas dedicados a fácil adaptação aos diferentes tipos de periféricos disponíveis, sejam eles entradas/saídas analógicas, entradas/saídas digitais, sistemas de barramentos seriais, etc. O ambiente de programação D'nia está sendo desenvolvido para integrar as características mencionadas acima.

3. Utilizando o D'nia

As raízes do D'nia podem ser encontradas no Instituto de Robótica do ETH em Zurique. O D'nia está relacionado em grande parte com a família de linguagens modulares Pascal, Modula-2 e Oberon desenvolvidas por N. Wirth. A última linguagem de programação de N. Wirth, Oberon [1,2], está ligado a um sistema operacional também chamado Oberon. O sistema Oberon completo se encontra disponível para PCs, Macintosh e muitas Workstations UNIX. D'nia utiliza Oberon System 3 de J. Gutknecht e H. Marais [5], como plataforma "host" para o desenvolvimento de software de tempo real.

Utilizando D'nia, o projetista de sistemas de tempo real pode programar em Oberon-2 ou C. A linguagem Oberon-2 [3,4] é muito apropriada para a programação em tempo real: ela é segura, modular, estruturada, limpa, legível, produz rapidamente código objeto e é simples e potente. A linguagem C dá acesso a bibliotecas padronizadas e a uma base enorme de programas em C, já existentes. Uma característica particular do D'nia é a simples e eficiente interface gráfica com o usuário.

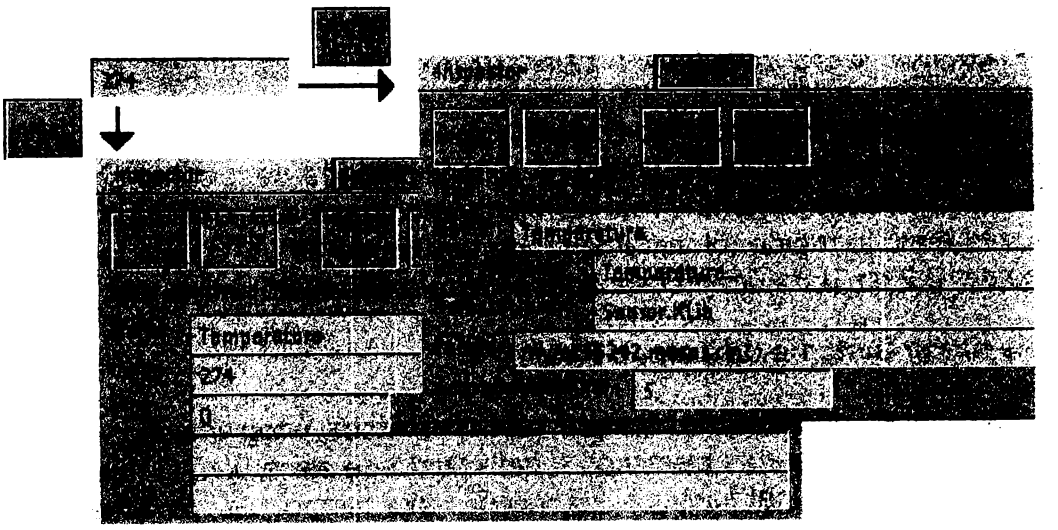

Figura 2: Exemplo de campo de saída de texto (no topo à esquerda contendo o valor 274). Este campo de texto mostra o valor atual de um sensor. O sensor está conectado ao sistema dedicado, e o campo de texto é apresentado no "host". É possível inspecionar (comando ***inspect****) este campo de texto de maneira a mostrar e definir atributos gráficos. Também é possível inspecionar o modelo (comando* ***inspect model****), o que fornece o endereço Internet, a biblioteca Oberon e o nome*

do objeto no sistema dedicado. Este método permite ao usuário conectar uma variável do programa no target com um objeto de visualização no "host".

É fácil criar novos botões, "check boxes", etc. e fazer o "link" destes com o código do sistema dedicado (veja figura 2). Desta forma um painel de controle de uma máquina, o "teach pendant" de um robô, ou outra interface qualquer para a aplicação de tempo real, pode ser facilmente criada em software.

O desenvolvimento propriamente dito se dá no PC ou em uma Workstation. O código é executado no sistema dedicado (target system). A comunicação entre o "host" e o "target" usa uma conexão serial ou ethernet com protocolo TCP/IP para assegurar uma comunicação confiável e padronizada. Num sistema multi-processado só é necessário uma única conexão física.

A chamada remota (feita no "host") de um comando contido em um módulo de programação não carregado no "target", inicia automaticamente um "down loading" e um "dynamic linking" do mesmo e dos respectivos módulos importados. Portanto, não é necessário nenhum comando para carregar e fazer o "link" dos módulos. O mecanismo de carregamento é implementado de forma que um módulo já carregado só possa ser substituído explicitamente por uma ação do usuário. Este conjunto de características proporciona uma enorme rapidez nos ciclos de edição, compilamento, carregamento e testes.

O sistema dedicado pode variar desde pequenas placas microcontroladoras até sistemas de grande porte do tipo VME com multi-processadores. Aplicações recentes foram feitas em um helicóptero autônomo utilizando sistema baseado no processador Motorola 68332, outra aplicação foi feita no controle de um AGV (autonomous guided vehicle) com 2 placas MVME162 e uma placa multi-processadora CNAPS-VME (veja também os exemplos a seguir).

A parte mais inovadora do D'nia é o sistema de tempo real orientado a objeto que comporta reutilização de código e extensibilidade sem compromissos com a eficiência. O sistema de tempo real foi escrito em quase sua totalidade em Oberon-2 e portanto é muito rápido (chaveamento de tarefas de 30 μs, e uma tarefa periódica mínima de 0.2 ms para um processador Motorola 68040/25) e portátil.

4. D'nia como Sistema de Tempo Real

Em geral núcleos de tempo real permitem programação em paralelo, sincronização entre tarefas paralelas e tratamento de exceções. O núcleo de tempo real dá ênfase principalmente às condições de tempo. As características mais importantes são:

Restrições de tempo e agendamento

Diferente de sistemas de tempo real convencionais que estabelecem prioridades para as tarefas críticas, o núcleo do D'nia define para cada tarefa de tempo crítico, valores de tempo. No núcleo (kernel) as condições de tempo devem ser atendidas utilizando os recursos do processador.

O núcleo comporta dois tipos de tarefas, tarefas de tempo crítico (TC) e tarefas de tempo não-crítico (NTC). Para as tarefas TC restrições de tempo precisam ser definidas. O planejador (scheduler) assegura que os recursos do processador sejam alocados para tarefas separadas. O planejador roda num intervalo fixo conhecido como base de tempo. Estes recursos são alocados de acordo com o seguinte esquema:

- Tarefas TCs tem prioridade;

- TCs podem consumir no máximo 90% dos recursos de processamento;

- Os recursos restantes não alocados às TCs são alocados para tarefas NTC.

Se uma tentativa é feita, para instalar uma nova TC, causando ao conjunto de tarefas TC a consumir mais do que 90% dos recursos, o sistema produz uma mensagem de alerta. Se nenhum recurso se encontra disponível o sistema gera uma situação excepcional.

Como foi mencionado acima as restrições de tempo estão associadas com tarefas TC. No caso de um controlador periódico, o período T e a duração D são definidos. A duração D deve ser determinado on-line. Dados D e T, a tarefa TC consumiria (D/T)*100% dos recursos disponíveis.

O tempo de que não é utilizado pelas tarefas TC é alocada para as tarefas NTC. Uma tarefa NTC é agendada dentro de "slots" como fatores de distribuição de tempo (DF), à seguir outra tarefa NTC é ativada formando um anel de tarefas NTCs.

A figura abaixo mostra um exemplo com quatro tarefas, duas das quais são tarefas periódicas: controller (ctrl) com período 1 e um planejador de trajetória (pp) com período 2 e outros dois processos NTC cmd1 e cmd2 com fator de distribuição 1. O planejador (scheduler) roda cada ocorrência de base de tempo, indicada pelas linhas verticais. A tarefa ctrl é feita executável todas as vezes, pp a cada duas vezes, e cmd1 e cmd2 recebem recursos de forma seqüencial.

Eventos e seus "handlers"

Trabalha-se no D'nia com eventos e seus "handlers". O "handler" é uma unidade de programação ou de tarefa que será ativada se um evento associado ocorre. Eventos podem ficar disponíveis através de uma série de fontes, um programa já em funcionamento pode gerar um evento e uma interrupção física pode de maneira semelhante gerar um evento .

Tabela de Tempo com o Evento "every"

O núcleo do D'nia define um evento fundamental every. Tarefas periódicas (TC) como controladores digitais são instalados como (novos) eventos "every" utilizando um evento preexistente como referência, e um fator de divisão de freqüência. Por motivos de eficiência e de sincronização entre as tarefas TC uma tabela de tempos com estrutura hierárquica é utilizada.

Eventos Externos

Uma característica importante de sistemas dedicados é a sua habilidade de reagir a sinais externos. Sinais típicos podem vir de chaves de fim de curso, recebimento de um caracter por um SIO, etc. Usualmente estes sinais causam uma interrupção do processador. O núcleo do D'nia serve automaticamente todas as interrupções do processador e gera um evento único para cada interrupção. O usuário pode portanto instalar seu próprio serviço de rotinas instalando um "handler" para o evento apropriado. Em outros casos (por exemplo switchs) nenhuma interrupção é gerada, entretanto, isto pode ser tratado de forma similar como se uma tarefa TC é instalada que recebe todos os registradores, gerando um evento apropriado.

Situações Excepcionais

Quando nenhum "handler" está presente para um dado evento (por exemplo uma interrupção SIO) diz-se que ocorreu uma situação excepcional. Atualmente para um evento comum, dispõe-se de um "handler" que não gera ação e este não pode ser substituído. Para todas as outras exceções o usuário pode fornecer o seu próprio "handler" de tratamento de execuções. Isto é útil para produtos de tempo real pois em muitos casos de emergência as máquinas precisam ser conduzidas automaticamente a um estado (posição) de segurança.

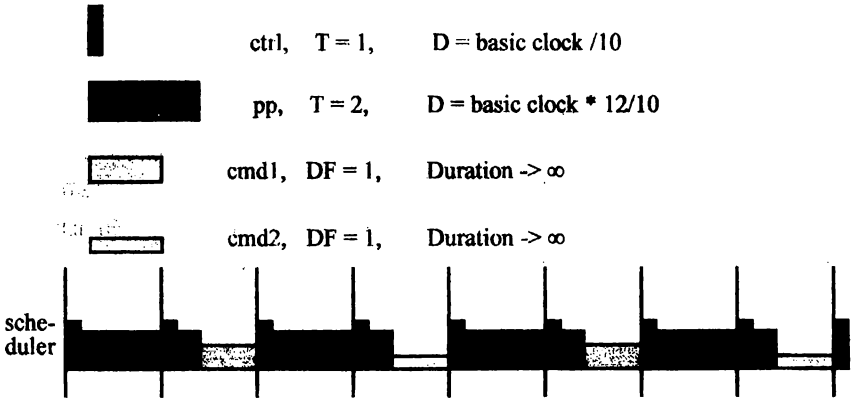

Fig. 3: Distribuição dos tempos de processamento entre tarefas através de um planejador (scheduler)

5. Ferramenta de Configuração

Para permitir o gerenciamento eficiente de periféricos, um mecanismo chamado de mecanismo de configuração foi elaborado para o D'nia. Por exemplo uma aplicação utiliza um sensor de força "*forceZ1*" que está conectado a um

conversor analógico-digital. Na inicialização da aplicação define-se a variável "z1" do sensor,

z1:=Define("forceZ1");

esta pode ser acessada por

val:=Get(z1)

resultando valores em Newtons. O offset, a escala e a localização exata do hardware do sensor é tratada pelo mecanismo de configuração. Isto significa, que a aplicação que utiliza "*forceZ1*" não está conectada a nenhuma das rotinas de driver, e portanto a reprogramaÁão ou recompilação da aplicação não é necessária no caso de alterações a nível de hardware. A conexão entre os identificadores definida no arquivo de configuração e a linguagem de programação é feita no momento da inicialização do sistema de controle. Um interpretador de comandos checa a configuração e instala as procedures com os parâmetros especificados.

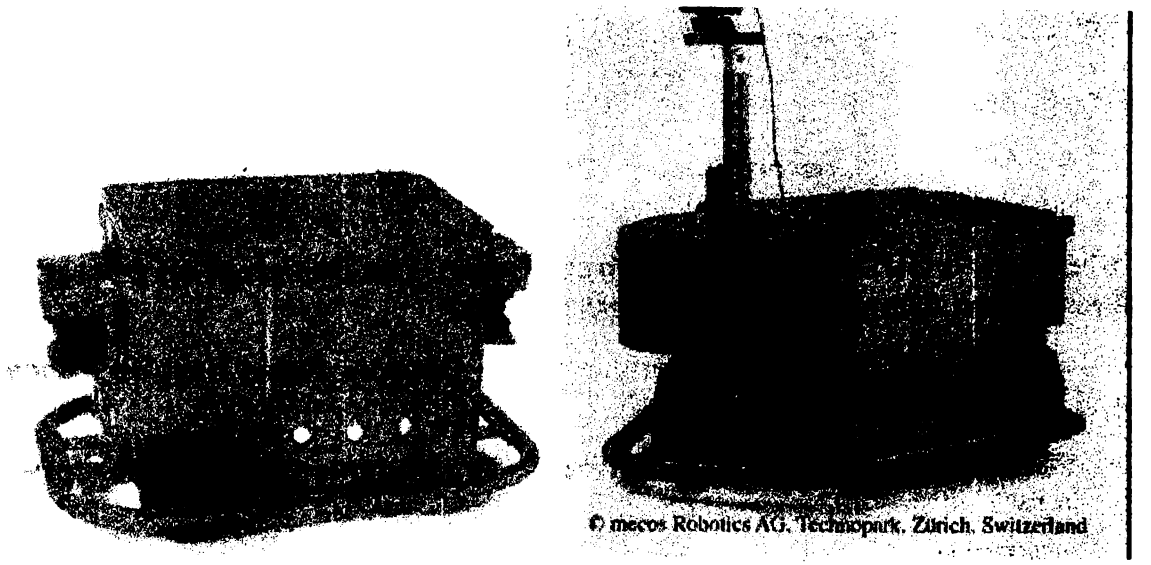

Figura 4: Dois AGVs da firma mecos Robotics AG. Ambos estão utilizando D'nia.

A implementação do mecanismo de configuração só foi possível devido a orientação a objeto do sistema. Uma biblioteca para placas VME, Módulos-IP e Modulos-M, barramento CAN e Interbus-S foram implementados.

6. Exemplos de Aplicações

A figura 4 mostra dois veículos-robô (AGV), o robô da esquerda está em operação na Escola Politécnica de Zurique (ETH) para pequisa em navegação autônoma em ambientes comuns de escritórios. O robô está equipado com controladores para seis eixos com dois processadores MVME162 rodando D'nia. A interface com as juntas servo-controladas é feita através de Módulos-IP e uma placa

proprietária para isolamento galvânico. Além disto, o robô tem 2 localizadores de distância laser interfaceadas com as 2 placas MVME162, usando RS485 a 115 kbit/s. Para segurança 12 sensores de distância ultra-sônicos foram integrados. As saídas analógicas destes sensores são multiplexadas por microcontroladores e os dados das medidas são transmitidos regularmente por um barramento CAN, um módulo-IP implementa a interface entre a placa MVME162 e o barramento CAN.

O AGV à direita é utilizado pela Universidade Técnica de Ilmenau na Alemanha. Este AGV adota um design cinemático com três rodas e uma configuração diferente de sensores. O principal sistema sensor deste AGV é uma câmera CCD. A imagem em tons de cinza da câmera é capturada por um "frame grabber" e transferida para um sistema multiprocessador CNPAS-VME para análise. O CNAPS é um sistema multi-processsador do tipo "single instruction multiple data", tendo em sua configuração 256 processadores. A performance de pico deste sistema é de $5x10^9$ multiplicações e acumulações/segundo. O CNAPS, o frame grabber e o próprio robô são controlados pelas duas placas MVME162. A conecção do computador host é realisada via ethernet.

7. Bibliografia

[1] REISER, M.. **The Oberon System. User Guide and Programmer's Manual**. Addison Wesley, 1991.

[2] WIRTH, N. & GUTKNECHT, J.. **Project Oberon. The Design of an Operating System and Compiler**. Addison Wesley, 1992.

[3] WIRTH, N. & REISER, M.. **Programming in Oberon. Steps beyond Pascal and Modula-2**. Addison Wesley, 1992.

[4] MÖSSENBÖCK, H.. **Objektorientierte Programmierung In Oberon-2**, Springer Verlag, 1993.

[5] GUTKNECHT, J.. Oberon System 3: Vision einer Softwaretechnologie der Zukunft. **Informatique**, 3, jun. 1994.