5º CONGRESSO BRASILEIRO DE PESQUISA E DESENVOLVIMENTO EM PETRÓLEO E GÁS

TÍTULO DO TRABALHO:

Simulação com Hardware In The loop Aplicada a Veículos Submarinos Semi-Autônomos

AUTORES:

Hilgad Montelo – hmontelo@gmail.com. Grupo de Sensores e Atuadores – USP.

Celso Massatoshi Furukawa - celso.furukawa@poli.usp.br. Grupo de Sensores e Atuadores – USP.

INSTITUIÇÃO:

Universidade de São Paulo - USP

Simulação com Hardware In The Loop Aplicada a Veículos Submarinos Semi-Autônomos.

Abstract

Unmanned Underwater Vehicles (UUVs) have many commercial, military, and scientific applications because of their potential capabilities and significant cost-performance improvements over traditional means of obtaining valuable underwater information The development of a reliable sampling and testing platform for these vehicles requires a thorough system design and many costly at-sea trials during which systems specifications can be validated. Modeling and simulation provide a cost-effective measure to carry out preliminary component, system (hardware and software), and mission testing and verification, thereby reducing the number of potential failures in at-sea trials. An accurate simulation environment can help engineers to find hidden errors in the UUV embedded software and gain insights into the UUV operation and dynamics. This work describes the implementation of a UUV's control algorithm using MATLAB/SIMULINK, its automatic conversion to an executable code (in C++) and the verification of its performance directly into the embedded computer using simulations. It is detailed the necessary procedure to allow the conversion of the models from MATLAB to C++ code, integration of the control software with the real time operating system used on the embedded computer (VxWORKS) and the developed strategy of Hardware in the loop Simulation (HILS). The Main contribution of this work is to present a rational framework to support the final implementation of the control software on the embedded computer, starting from the model developed on an environment friendly to the control engineers, like SIMULINK.

Introdução

Testes tradicionais, muitas vezes referenciados como testes estáticos, consistem da avaliação de funcionalidades de um componente particular onde a ele são fornecidas entradas conhecidas e saídas mensuráveis. O aumento da pressão pelo lançamento de produtos mais rapidamente ao mercado e o apelo pela redução dos ciclos de desenvolvimento associados ao projeto têm ocasionado um aumento da necessidade de testes dinâmicos, onde o comportamento dos componentes é avaliado à medida que são submetidos a interações com o restante do sistema, ora de forma real ou simulada. Testes dinâmicos minimizam riscos relacionados à segurança e custos, além de abranger maiores condições de testes quando comparados aos testes estáticos. A aplicação desta estratégia para testes dinâmicos é conhecida como simulação com hardware in the loop (*HIL*).

A simulação com hardware in the loop surge como uma ferramenta que vem para facilitar o trabalho do engenheiro de controle. Sistemas podem ser desenvolvidos rapidamente utilizando apenas o modelo conceitual por um único engenheiro de controle. Neste ambiente de desenvolvimento, o engenheiro de controle pode focar-se inteiramente sobre características do projeto, evitando levar horas de escrita e/ou depuração de códigos ou mesmo ter que lidar com as complexidades inerentes aos sistemas de tempo real. Eventualmente o projeto precisará ser transferido para o sistema "alvo" (*target*). Desde que o algoritmo de controle seja conhecido, passa a ser conhecida também sua carga computacional, podendo-se portanto selecionar um processador adequado para a execução das tarefas.

A técnica de simulação com hardware in the loop aplica-se a todos os sistemas, sejam eles grandes ou pequenos, processos industriais e mesmo durante o desenvolvimento de novos produtos. Onde quer que exista interação entre simulação e o mundo real, existe uma oportunidade para a abordagem de simulação com hardware in the loop. Isto permite a utilização de *HIL* durante o projeto e desenvolvimento de um Veículo Submarino Não Tripulado (UUV - *Unmanned Underwater Vehicles*), equipamento composto de uma coleção de complexos subsistemas (navegação, controle, atuadores, alimentação, etc) todos integrados dentro de uma plataforma embarcada, provendo os recursos necessários para suprir as necessidades que levam à construção de veículos cada vez mais complexos e autonômos.

Metodologia

Plataforma de Desenvolvimento

A plataforma de desenvolvimento consiste, especificamente, da montagem e utilização de hardware com características semelhantes às apresentadas pelo projeto do veículo submarino que se encontra em desenvolvimento nos laboratórios da Engenharia Mecatrônica da Escola Politécnica de São Paulo (LIMA, et al., 2003), (LIMA, 2003a), (ALMEIDA, et al., 1999), permitindo o aproveitamento de recursos e soluções em pesquisa com temas relacionados.

Ambiente Operacional de Tempo Real

VxWorks é um sistema operacional de tempo real fabricado pela Windriver Systems. Como a maioria dos sistemas operacionais de tempo real (VXWORKS, 2005). Ele inclui um kernel com suporte a multi-tarefas e escalonamento preemptivo. Muito popular em aplicações embarcadas é utilizado em varias arquiteturas de hardware, incluindo MIPS, PowerPC, SH-4, x86, ARM, StrongARM, xScale.

Framework de Desenvolvimento em Sistemas de Tempo Real

O Constellation, consiste de um framework de tempo real orientado a objetos (CONSTELLATION, 2005), com capacidade de interfaceamento e geração de código partindo de um modelo projetado em MATLAB/Simulink até a obtenção de código em linguagem de programação nos padrões ANSI C++. Favorece a componentização de software, método no qual é elevada a taxa de reutilização e especialização de código.

Dinâmica do UUV

Um dos primeiros passos no desenvolvimento de uma simulação apurada é a modelagem das equações dinâmicas do Hornet UUV (BRAUNL, et al., 2004). Neste caso, as coordenadas referentes ao corpo rígido são descritas pelos seguintes 6 (seis) graus de liberdade e suas respectivas derivadas, conforme mostra a Figura 1 (x - Movimento linear com relação ao eixo longitudinal; y - Movimento linear com relação ao eixo transversal; z - Movimento linear com relação ao eixo vertical; φ - Movimento rotacional sobre o eixo X; θ - Movimento rotacional sobre o eixo Y e ψ - Movimento rotacional sobre o eixo Z).

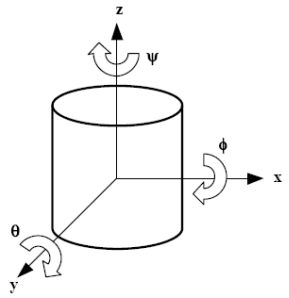

Figura 1. Demonstração de corpo rígido com sistema de coordenadas.

As entradas do sistema são definidas da seguinte forma: $\mathbf{T}_1$- Força propulsora aplicada pelo propulsor lateral; $\mathbf{T}_2$- Força propulsora aplicada pelo propulsor frontal; $\mathbf{T}_3$- Força propulsora aplicada pelo propulsor vertical; $\mathbf{F}_{ext(x,y,z,\psi)}$ – Distúrbios externos (correnteza da água, etc..); $\mathbf{F}_{d(x,y,z)}$ – Forças de arrasto hidrodinâmico linear devido ao movimento do veículo definido pela equação 1 (ROBERSON, et al., 1997):

$$\mathbf{F}_d = \mathbf{C}_d * \rho * \mathbf{A}_p * \frac{\mathbf{V}_0^2}{2} \qquad (1)$$

Onde:

$$C_d = Coeficiente\ de\ arrasto$$

$\rho = densidade\ da\ água$

$A_p = Área\ de\ arrasto\ projetada$

$V_0 = Velocidade\ da\ superfície\ de\ arrasto$

Outros parâmetros importantes:

$m - massa\ do\ veículo = 90\ kg$

$D - Separação\ entre\ os\ propulsores\ T_1\ e\ T_2 = 0{,}28\ m$

$W_{auv} - Peso\ do\ veículo\ na\ água\ (considerando\ empuxo\ 0)$

I_z - Momento de inércia sobre o eixo-z = $0{,}004\ kg * m^2$

As equações dinâmicas de movimento podem ser derivadas da seguinte maneira (OLGAC, et al., 1991). Tomando a soma de todas as forças nas direções dos eixos X, Y e Z e resolvendo as respectivas acelerações apresentadas nas equações 2, 3 e 4, respectivamente:

$$\ddot{x} = \frac{(T_1 + T_2) + F_{dx} + F_{xext}}{m} + \dot{y}\,\dot{\psi}, \quad (2) \qquad \ddot{y} = \frac{F_{dy} + F_{yext}}{m} + \dot{x}\,\dot{\psi}, \quad (3) \qquad \ddot{z} = \frac{T_3 + Tdz_2 - W_{auv} + F_{zext}}{m}, \quad (4)$$

onde:

$F_{d(x,y,z)}$ são definidas pela equação 1.

$F_{xext} = F_{yext} = F_{zext} = 0$ (assumindo nenhuma correnteza externa ou distúrbio).

Para os propósitos da simulação, estas equações de movimentos comportam 8 (oito) variáveis de estado e 3 (três) entradas de sistemas independentemente controladas, apresentadas na equação 5, a seguir:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix}, \qquad u = \begin{bmatrix} F_z \\ M_z \\ F_z \end{bmatrix} = \begin{bmatrix} T_1 + T_2 \\ (T_2 - T_1)\left(\frac{D}{2}\right) \\ T_3 \end{bmatrix}, \qquad (5)$$

onde u é em função de T_1, T_2 e T_3 (as entradas para o modelo da planta dos três controladores de propulsão).

Implementando um ambiente de Simulação com Hardware in the loop

A Figura 2 apresenta uma visão geral da arquitetura de hardware in the loop utilizada para auxiliar no desenvolvimento e construção do veiculo submarino autônomo. Possuindo os seguintes componentes: **Hardware embarcado -** Representa parte de hardware utilizado pelo Veículo Submarino Semi-Autônomo real. Suas entradas consistem das forças vetoriais geradas pelo meio (correntes, empuxo, etc) e o vetor de torque aplicado pelos propulsores e planos de controle; **Modelo do Mundo -** Consiste do modelo do mundo físico no qual será inserido veículo, possuindo a representação de diferentes fenômenos físicos como correntezas e ondas, por exemplo; **Parâmetros de controle -** Variáveis principais e auxiliares usadas para a realização adequada dos algoritmos de controle empregados no veículo, como velocidade, aceleração, gravidade, forca empuxo, sentido e direção da correnteza; **Sensores e atuadores -** Componentes de entrada e saída do ambiente, respectivamente. Podem ser compontes reais (preferencialmente o mesmo hardware adquirido pelo veículo submarino) ou virtuais (arquivos de entradas e/ou saídas); **Data Logger**. Registro das estruturas de dados obtidos durante testes e operação, sejam eles total ou parcialmente simulados. Através deste componente é possível rever dados da operação, validar e verificar a implementação dos algorítimos de controle.

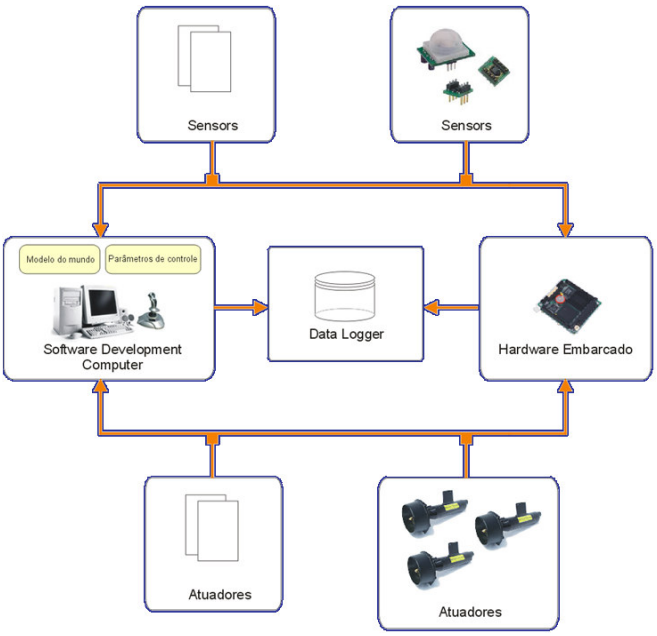

Figura 2. Visão geral da arquitetura de simulação com hardware in the loop aplicada ao desenvolvimento e construção de Veiculo Submarino Autônomo.

Os seguintes passos são necessários para gerar código compatível com a arquitetura de hardware in the loop proposta, baseado, é claro, inicialmente no desenvolvimento e análise do modelo conceitual, lógico e físico dos problemas associados à construção de um veiculo submarino autônomo: Preparar o modelo do controlador em Simulink para o UUV; Converter aquele modelo em um componente de software compatível com o modelo e arquitetura do *Constellation*; Preparar o ambiente target com sistema operacional de tempo real VxWorks; Configurar o *Constellation* para gerar código correspondente para a máquina target ou para ambiente de simulação (código em C/C++ sem considerar as restrições de tempo gerenciadas pelo S.O. de tempo real); Outro recurso ainda utilizado é a integração com a ferramenta Stethoscope® da WindRiver (SCOPETOOLS, 2005), que fornece gráficos de acompanhamento dos valores de variáveis que deseja-se monitorar.

Resultados e Discussão

O modelo dinâmico do UUV e seu algoritmo de controle, publicados em (BRAUNL, et al., 2004) foram reproduzidos em MATLAB/Simulink com sucesso. Simulações feitas em MATLAB/Simulink reproduziram com fidelidade o comportamento do UUV descrito na literatura.

Modelo Dinâmico do UUV adaptado para execução no ambiente VxWorks

Foi necessário realizar adaptações no modelo dinâmico original do UUV para eliminar loops algébricos do modelo em Simulink fornecido pelos autores do Hornet que impediam a sua conversão para código executável em VxWorks. Esses loops são causados por realimentação direta, em que uma saída é direcionada à entrada do mesmo bloco, ou quando um bloco é realimentado por um outro bloco que também tem realimentação.

Simulações com o controlador embarcado

Os gráficos apresentados a seguir procuram mostrar que o sistema de controle embarcado no PC104 consegue gerar os sinais de atuação em conformidade com os resultados encontrados na literatura. Ou seja, o controlador implementado em MATLAB/Simulink foi portado com sucesso para o PC104. Os gráficos foram gerados na seguinte sequência: Amostras dos sinais representando os sensores foram gravados em arquivos, que foram transferidos para o sistema de arquivos do PC104; O

código do controlador (embarcado no PC104) leu as amostras dos sinais e calculou os sinais de atuação dos propulsores, segundo o algoritmo de controle implementado; Os sinais de saída que seriam enviados aos propulsores no UUV foram gravados em arquivos no PC104; Por fim, esses arquivos de atuação foram alimentados no simulador dinâmico do UUV, reproduzindo os gráficos de deslocamento e velocidade (curvas indicadas com o rótulo "Reproduzido"), que foram plotados juntamente com as curvas geradas pelo simulador original.

Simulação 1: Descolamento em direção a um ponto específico

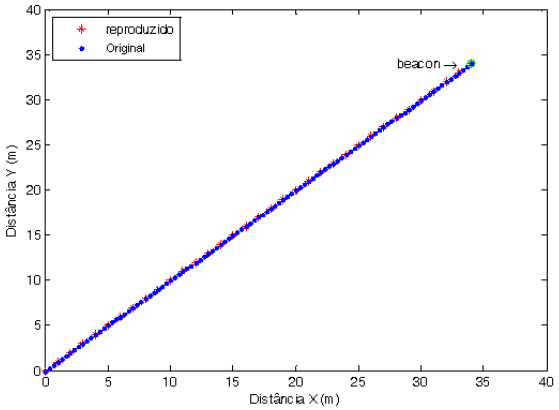

Figura 3. Visualização do caminho percorrido pelo UUV.

Do ponto de partida (0,0), o UUV percorre o mapa por ele conhecido até encontrar o ponto indicado (target), Figura 3 , que encontra-se, naquele momento, nas coordenadas (35,35) do mapa. A distância percorrida pelo UUV é de aproximadamente 49 m. Note que o resultado é compatível com o resultado publicado pelos autores do Hornet em (BRAUNL, et al., 2004),e o mesmo pode-se dizer para os resultados apresentados a seguir.

Simulação 2: Controle de Profundidade

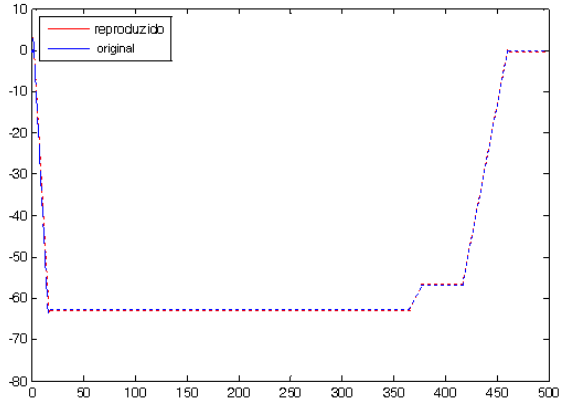

Figura 4. Percurso em profundidade empreendido pelo UUV.

Na simulação apresentada pela Figura 4, o veículo navega até a localização de seu alvo. Durante esta missão, o veículo foi comandado para descer em duas diferentes profundidades, uma a 64 metros e depois sobe a 55 metros, e a uma distância de 30 metros do alvo, o veículo sobe a 1 metro da superfície. O controlador PID de profundidade comportou-se bem como um controlador PD com $k_p = 10$, $k_i = 0$ e $k_d = 5$.

Simulação 4: Controle de Velocidade

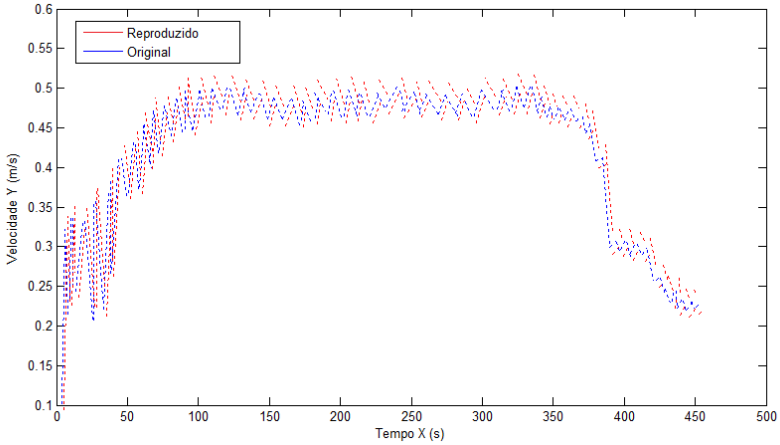

Figura 5. Performance do Controlador de velocidade PID.

O controlador de velocidade PID, integrado ao controlador de deslocamento, apresentou dificuldades em manter a velocidade, como mostra a Figura 5, porque foi constantemente chaveado em favor das correções de erros. Apesar disso, o UUV permaneceu seguindo em frente. Uma vez mais o comportamento apresentado pelo controlador embarcado apresenta comportamento similar aos resultados originalmente publicados para o mesmo modelo de UUV.

Integridade do contexto em ambiente de tempo real

Pela Figura 6 é possível observar que o ciclo de controle implementado, encapsulado pela tarefa CSSysClockData e indicado na circunferência em vermelho, não excedeu o seu período de tempo permitido de execução, na realidade, considerando que o período é de aproximadamente 15 ms (indicado pelas barras vermelhas) e que a execução do módulo de controle só leva 2 ms para executar suas rotinas, significa dizer que a implementação do algoriítmo de controle consome menos de 15% do tempo disponível. Dessa forma o sistema de hardware utilizado possui capacidade de processamento suficiente para atender o algoritmo de controle implementado.

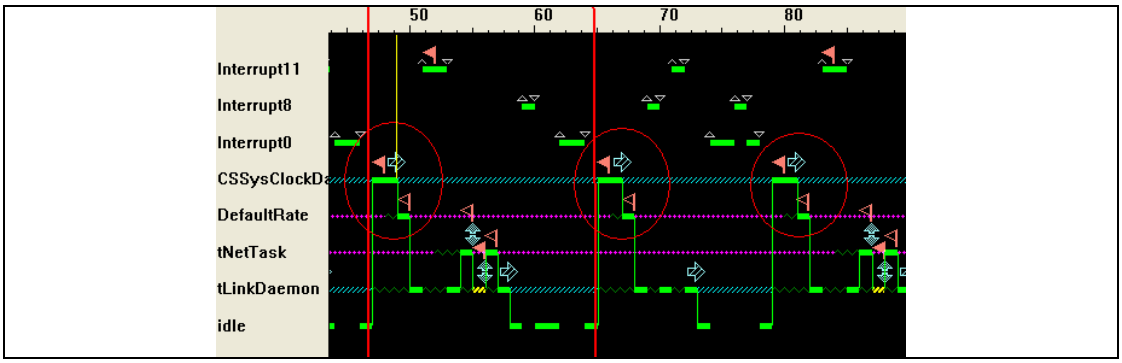

Figura 6. Visualização de tempo de execução e troca de contexto entre tarefas.

Este tipo de análise permite avaliar o relacionamento existente entre o hardware a ser utilizado no UUV e o algoritmo de controle implementado, buscando uma forma de compatibilizá-los de acordo com o resultado obtido.

Conclusões

O Modelo dinâmico e controlador do UUV Hornet foi reproduzido com sucesso em ambiente MATLAB/Simulink. O resultados das simulações são compatíveis com o da literatura disponível,

mesmo após a realização de alguns ajustes, como a adição de alguns atrasos para evitar a formação de *loops* algébricos - uma limitação do próprio ambiente MATLAB/Simulink que impede a conversão de código em C++.

A simulação com hardware in the loop funcionou de acordo com o esperado, comportando-se adequadamente nas etapas de leitura de um modelo de controle de um UUV, conversão daquele modelo em código executável e validação do algoritmo implementado em ambiente real. O controlador embarcado mostrou comportamento muito similar ao disponível na literatura e atendeu às restrições de tempo e processamento impostas pelo hardware utilizado.

Pode-se verificar a integridade de contexto com o software de controle rodando no PC104 por meio de ferramenta do ambiente VxWorks. Trata-se de um funcionalidade bastante útil, porque através dela o engenheiro de controle pode avaliar se o algoritmo em desenvolvimento está compatível com as especificações do hardware utilizado como computador embarcado; análise que, dependendo do resultado, pode levar a uma melhoria do algoritmo (em termos de performance) ou mesmo apontar a necessidade de se trocar o hardware por outro com melhor desempenho .

Com a utilização da estrutura montada para a realização das simulações com *hardware in the loop*, parte das complexidades inerentes ao se lidar com sistemas operacionais de tempo real (sincronização, escalonamento, prioridades, etc), são tratadas de forma quase transparente, de forma a ajudar o projetista ou engenheiro de controle a ter um foco maior sobre o problema.

Problemas encontrados durante o envio/recepção de comandos e atualização remota de bibliotecas sugerem o desenvolvimento de um modelo de carga computacional com enfoque na comunicação com processos e eventos externos de forma que seja possível avaliar melhor o comportamento dos algoritmos de controle implementados na ocorrência dos mesmos.

Agradecimentos

À minha família e amigos pelo incentivo permanente e torcida pelo sucesso na conclusão dessa dissertação; À FINEP que através de um de seus fundos setoriais, o CT-PETRO, possibilitou parte do finaciemento deste trabalho; Ao CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, por haver concedido a bolsa de estudos para a realização deste mestrado; À Universidade de São Paulo e seus colaboradores - que introduziu este programa de pós-graduação, inovando no ambiente do ensino acadêmico.

Referências Bibliográficas

(ALLES, et al., 1994) **ALLES S., SWICK C. e HOFFMAN M. et al** "A Real Time Hardware in the Loop Simulation for Traction Assist". International Journal of Vehicle Design, 1994. Vols. 15, pp. 597-625.

(ALMEIDA, 1999) **ALMEIDA P. E. M., SIMÕES, M. G.** "Projeto de um sistema robótico inteligente para instalação de equipamentos em poços petrolíferos em águas profundas". São Paulo : IV SBAI – Simpósio Brasileiro de Automação Inteligente, 1999.

(ANAKWA, et al., 2002) **ANAKWA W. K. [et al.]** "Environments for rapid implementation of control algorithms and hardware in the loop simulation". In IEEE 28th Annual Conference of the Industrial Electronics Society, 2002.

(BRAUNL, et al., 2004) **BRAUNL T. [et al.]** "The Autonomous Underwater Vehicle Initiative – Project Mako". Singapore : IEEE Conference on Robotics, Automation, and Mechatronics (IEEE-RAM), 2004. Vols. pp: 446-451.

(BRUTZMAN, et al., 1992) **BRUTZMAN D. P., KANAYAMA Y. e ZIDA M. J** "Integrated Simulation for Rapid Development of Autonomous Underwater Vehicles". IEEE Symposium On Autonomous Underwater

Vehicle Technology., 1992.

(CONSTELLATION, 2005) **CONSTELLATION** "Real Time Innovations - Constellation, Real-Time Software Framework ". Real-Time Innovations, Inc, 2005.

(DUNO, et al., 1994) **DUNO S. E., SMITH S. M. e BETZER P.** "Design of Autonomous Underwater Vehicles for Coastal Oceanography, Underwater Robotic Vehicles. Design and Control". TSI Press, 1994. Vols. pp. 229-326.

(JANSSON, et al., 1994) **JANSSON, A. e PALMBERG, J. O**. "Load simulation, a flexible tool for assessing the performance of hydraulic valves". In Proceedings of the Fourth Triennial International Symposium on Fluid Control, Fluid Measurement, and Visualisation, Toulouse, France, 1994.

(KEY, 1987) **KEY C.** "Cooperative Planning in the Pilot's Associate". Proc. DARPA Knowledge-Based Planning Workshop., 1987.

(KIM, et al., 2002) **KIM J. e SRINIVAN M. A.** "Computationally efficient technique for real time surgical simulation with force feedback". Proc. 10th Sym. On Haptic Interfaces For Virtual Environment & Teleoperator Systems, 2002.

(LIMA, et al., 2003) **LIMA, F. V. F.; FURUKAWA, C. M**. "Development of a High Resolution Underwater Positioning System". XIV COBEM – Congresso Brasileiro de Engenharia Mecânica. 2003.

(LIMA, 2003a) **LIMA, F. V. F**." Desenvolvimento de um Sistema Acústico de Posicionamento Submarino". Dissertação de mestrado apresentada à Escola Politécnica da USP. São Paulo. 2003.

(MACLAY, 1997) **MACLAY D.** "Simulation Gets Into the Loop". IEEE Review, 1997.

(OLGAC, et al., 1991) **OLGAC N., PLATIN B.E. e CHANG J. M.** "Sliding Mode Control of Remotely Operated Vehicles for Horizontal Plane Motions.". IEEE Proceedings, 1991. Vol. 138.

(PRASETIAWAN, et al., 1999) **PRASETIAWAN E. A. [et al.]** "Modeling and control design of a powertrain simulation testbed for earthmoving vehicles". Journal of Fluid Power Systems and Technology, 1999.

(ROBERSON, et al., 1997) **ROBERSON J. A. e CROWE C. T.** "Engineering Fluid Mechanics". New York : John Wiley & Sons, 1997. Vol. Sixth Edition.

(SCOPETOOLS, 2005) **SCOPETOOLS** "Windriver's ScopeTools Suite". Windriver Corp., 2005.

(SENTA, et al., 2002) **SENTA Y e OKAMURA E.** "HIL simulation system for hdd servo firmware". IEEE Transactions on Magnetics, 2002. pp: 2204-2207.

(VXWORKS, 2005) **VXWORKS** "Windriver - VxWorks Programmer's Guide". Windriver, 2005. Vol. 5.5.

(YURODA, et al., 1996) **YURODA Y., ARAMAKI K. e URA T.** "AUV Test Using Real/Virtual Synthetic World". IEEE Symp. On Autonomous Underwater Vehicle Technology, 1996.

(ZHANG, et al., 2003) **ZHANG R., CARTER D. E. e ALLEYNE A. G.** "Multivariable control of an earthmoving vehicle powertrain experimentally validated in an emulated working cycle". Washington : In Proc ASME International Mechanical Engineering Congress & Exposition, 2003.

(ZHEN, et al., 2004) **ZHEN L., KYTE M. e JOHNSON B.** "Hardware in the loop real-time simulation interface software design". In Proc of The 7th International IEEE Conference on Intelligent Transportation Systems, 2004. pp: 1012-1017.