sid.inpe.br/mtc-m21d/2021/06.23.01.52-TDI

ESTUDO DA MIGRAÇÃO E IMPLEMENTAÇÃO DE UM SOFTWARE DE CONTROLE DE ATITUDE PARA UM SISTEMA OPERACIONAL E UM PROCESSADOR EM TEMPO REAL.

Michel Macena Oliveira

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 12 de maio de 2021.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/44TKTBS>

INPE
São José dos Campos
2021

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

sid.inpe.br/mtc-m21d/2021/06.23.01.52-TDI

ESTUDO DA MIGRAÇÃO E IMPLEMENTAÇÃO DE UM SOFTWARE DE CONTROLE DE ATITUDE PARA UM SISTEMA OPERACIONAL E UM PROCESSADOR EM TEMPO REAL.

Michel Macena Oliveira

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 12 de maio de 2021.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/44TKTBS>

INPE
São José dos Campos
2021

Dados Internacionais de Catalogação na Publicação (CIP)

Oliveira, Michel Macena.

Ol4e Estudo da migração e implementação de um software de controle de atitude para um sistema operacional e um processador em tempo real. / Michel Macena Oliveira. – São José dos Campos : INPE, 2021.

xxviii + 136 p. ; (sid.inpe.br/mtc-m21d/2021/06.23.01.52-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2021.

Orientador : Dr. Marcelo Lopes de Oliveira e Souza.

1. Simulação. 2. Controle. 3. Tempo real. 4. RTEMS. 5. Processadorna- Malha. I.Título.

CDU 629.7.062.2:004.4

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

DEFESA FINAL DE DISSERTAÇÃO DE MICHEL MACENA OLIVEIRA
BANCA Nº 087/2021, REG 141968/2018

No dia 12 de maio de 2021, às 10h00min, por teleconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi APROVADO(A) pela Banca Examinadora por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle. O trabalho precisa da incorporação das correções sugeridas pela Banca Examinadora e revisão final pelo(s) orientador(es).

Novo título: "ESTUDO DA MIGRAÇÃO E IMPLEMENTAÇÃO DE UM SOFTWARE DE CONTROLE DE ATITUDE PARA UM SISTEMA OPERACIONAL E UM PROCESSADOR EM TEMPO REAL"

Eu, Mário César Ricci, Presidente da Banca Examinadora, assino esta ATA, em nome de todos os membros, com o consentimento dos mesmos.

Membros da Banca

Dr. Mário César Ricci - Presidente – INPE
Dr. Marcelo Lopes de Oliveira e Souza – Orientador - INPE
Dr. Renato Oliveira de Magalhães - Membro Interno - INPE
Dr. Fernando Antonio Pessotta – Membro Interno – INPE
Dr. Fernando José de Oliveira Moreira - Membro Externo - EMBRAER

Documento assinado eletronicamente por **Mario Cesar Ricci**, **Tecnologista**, em 17/05/2021, às 18:16 (horário oficial de Brasília), com fundamento no art. 6º do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.mctic.gov.br/verifica.html, informando o código verificador **7087964** e o código CRC **88036BEB**.

AGRADECIMENTOS

Agradeço ao meu pai Jonas Correa Oliveira por me apresentar a área técnica, incentivar aos estudos e me dar apoio mesmo em condições difíceis.

Agradeço ao meu orientador da graduação Alexandre Carvalho Leite pelo incentivo ao mestrado e por me indicar o INPE.

Agradeço ao meu orientador Marcelo Lopes de Oliveira e Souza por me orientar e me ensinar a enxergar e construir inúmeros conceitos e ideias em diferentes áreas e pela sua incansável vontade de ensinar.

Agradeço aos meus amigos Herbi Junior Pereira Moreira, José Vieira da Silva Neto e Alexandre Dias Luciano Lima pela amizade e suporte, sem os quais haveria sido muito mais difícil realizar este trabalho.

Agradeço à Palmira dos Santos por me incentivar a entrar no INPE, me receber em São José dos Campos e a encontrar um local para morar.

Agradeço aos servidores engs. Fabrício de Novaes Kucinskis e Fernando Pessota pelo apoio técnico e disponibilidade do Kit TSC695 sem o qual a realização deste trabalho seria severamente prejudicada.

Finalmente agradeço ao INPE, à CAPES e aos professores do Curso ETE/CMC pelos ensinamentos e condições para a realização deste trabalho.

RESUMO

Atualmente, sistemas de satélites são comuns no mundo, seja para propósitos civis ou militares. Mas sistemas de satélites demoram muito para construir, operar e se pagar, são de altíssimo custo e complexidade. Suas falhas geram riscos de perda da missão e prejuízos da ordem de milhões de reais/dólares. O segmento espacial apresenta alto custo; porém, o planejamento e realização da missão também têm altos custos que devem ser considerados numa análise de riscos. Portanto, há extrema importância em métodos e meios para evitar tais falhas, como projetar: 1) o segmento espacial para suportar as intempéries do espaço; 2) A missão para suportar as demandas operacionais; e 3) todos, diante de um conjunto de falhas considerado. Particularmente, um satélite é composto de diversos subsistemas; e cada um contribui para o risco de falhas à sua maneira. Um dos mais influentes é o Sistema de Controle de Atitude e de Órbita (SCAO) por sua complexidade e operação em tempo real. Por isso, este trabalho estuda a implementação de um software de controle de atitude para um sistema operacional e um processador em tempo real. O controle de órbita não é tratado. Para isto, ele: 1) recupera um ambiente e um software de SCA já desenvolvido para o sistema operacional em tempo real RTEMS que executa sobre um emulador do processador ERC32 e se comunica com um computador PC que simula o ambiente espacial; 2) Migra-os para um ambiente semelhante; porém, os implementa sobre um processador ERC32 real; 3) os simula e os compara nos mesmos casos e nas mesmas condições dos desenvolvimentos anteriores. Este método permite: 1) aproveitar e continuar os desenvolvimentos anteriores; 2) recuperar e melhorar tal ambiente e tal software; 3) testar o software de SCA em tempo de projeto, sob diversas condições que poderão ocorrer em tempo de execução da missão, antes de embarcá-lo no modelo de vôo de um satélite; e assim, 4) reduzir os riscos de falhas e perda da missão. Os resultados obtidos e tais comparações sugerem o sucesso de tal migração.

Palavras-chave: Simulação. Controle. Tempo Real. RTEMS. ERC32. Processador-na-Malha.

STUDY OF THE MIGRATION AND IMPLEMENTATION OF AN ATTITUDE CONTROL SOFTWARE FOR A REAL TIME OPERATIONAL SYSTEM AND PROCESSOR

ABSTRACT

Currently, satellite systems are common in the world, whether for civil or military purposes. However, satellite systems take long time to build, operate, and pay, are very costly and complex. Their faults generate risks of mission loss; as a result, millions of reais / dollars losses. The space segment has a high cost; but mission planning and realization also have high costs that should be considered in a Risk Analysis. So, there is extreme importance in methods and means to avoid such faults, as designing: 1) the space segment to support space conditions; 2) the mission to support the operational demands; and 3) all subject to a considered set of faults. In particular, a satellite is made up of several subsystems; and each contributes to such faults in its own way. One of the most influentials is the Attitude and Orbit Control System (AOCS) for its complexity and real time operation. Thus, this work studies the implementation of an attitude control software to a real time operational system and processor. The orbit control is not treated. To do this, it : 1) recovers an environment and ACS software already developed for the RTEMS real-time operating system, running over an ERC32 processor emulator, and communicating with a PC computer that simulates the space environment; 2) migrates them to a similar environment; but, implements them on an actual ERC32 processor; 3) simulates them and compare them in the same cases and under the same conditions of the previous developments. This method allows to: 1) avail and continue the previous developments; 2) recover and improve such environment and software; 3) test the ACS software at design time, under various conditions that may occur in mission execution time, before embed it on a satellite flight model; and thus 4) reduce the risks of faults or loss of mission. The results obtained and such comparisons suggest the success of such migration.

Keywords: Simulation. Control. Real Time. RTEMS. ERC32. Processor-In-The-Loop.

LISTA DE FIGURAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

ERC32	–	Embedded Real-time Computing core - 32 bits
SCA	–	Sistema/Software de Controle de Atitude
PMM	–	Plataforma MultiMissão
SOTR	–	Sistema Operacional em Tempo Real
RTEMS	–	Real-Time Executive for Multiprocessor Systems
MPS	–	Multi-Processamento Simétrico
ESA	–	European Space Agency
SPARC	–	Scalable Processor Architecture
UPF	–	Unidade de Ponto Flutuante
UI	–	Unidade de Inteiros
CM	–	Controlador de Memória(CM)
RI	–	Referencial Inercial
RO	–	Referencial Orbital
RS	–	Referencial do Satélite
VLHL	–	Vertical Local Horizontal Local
ERC32CSS	–	ERC32 GNU Cross-Compiler System
SIS	–	Sparc Instruction Simulator
UART	–	Universal Asynchrounous Receiver/Transmiter
PIL	–	Processor-In-The-Loop
PIC	–	Programmable Intelligent Computer (Microchip)
DSP	–	Digital Signal Processor
DEA	–	Divisão de Eletrônica Aeroespacial

SUMÁRIO

Pág.

1 INTRODUÇÃO

Satélites artificiais são veículos espaciais que operam em um ambiente completamente hostil. A perda de um satélite implica prejuízo da ordem de milhões de dólares para uma nação. Isto significa que cada passo do projeto de uma missão espacial é extremamente crítico, pois um erro conceitual, um descuido ou uma falha da equipe de projeto pode acarretar a perda total da missão. Portanto, é necessário realizar uma exaustiva análise da missão, inúmeros testes e simulações a fim de se mitigar o tanto quanto possível a probabilidade de falha da missão. No INPE, e para Sistemas/Softwares de Controle de Atitude (SCA) destaca-se a longa sequência de trabalhos desde Souza (1981), passando sobretudo por Gobato (2006), Moreira (2006), Amorim III (2009), até Tagawa (2013),Moraes (2017), Santos (2020) e o presente trabalho. Este trabalho é um continuação direta do trabalho de Amorim III (2009) e preserva sua abordagem de modelagem e simulação.

O trabalho de Amorim III (2009) consistiu em aumentar o grau de fidelidade do simulador do Modo Nominal de funcionamento de um SCA para modelos similares aos da Plataforma MultiMissão (PMM) de 2001, com vários controladores calculados no trabalho de desenvolvido no trabalho de Gobato (2006) e com software desenvolvido no trabalho de Moreira (2006). Este desenvolveu um simulador que imita o Software de Controle de Atitude (SCA) para modelos similares aos da PMM de 2001 e o "Mundo Externo"ao SCA (dinâmica do satélite e ambiente espacial). Amorim III (2009) reorganizou este simulador e o implementou sobre um software simulador da arquitetura de um processador ERC 32 em diferentes condições. Este processador foi desenvolvido para uso em satélites e o mesmo se encontra no projeto da PMM.

O presente trabalho visa recuperar e preservar o simulador reorganizado por Amorim III (2009) e aumentar novamente o grau de fidelidade do simulador. Desta forma o mesmo fica mais adequado à realidade física do sistema. A abordagem atual consiste em migrar o SCA para outro ambiente. Neste caso, o SCA é implementado sobre um processador ERC32 real situado em uma plataforma (placa eletrônica) com interface de comunicação serial. O software "Mundo Externo"é implementado sobre um computador PC regular com sistema operacional Linux. Esta abordagem permite avaliar o software do SCA diretamente implementado sobre um processador para uso em satélites em tempo de projeto. Portanto, isto contribui para a diminuição do risco de falhas do software embarcado em tempo de execução (voo).

1.1 Motivação e justificativa

Satélites são sistemas de alto custo e complexidade;

- Portanto suas falhas geram riscos até de perda da missão ou do satélite e prejuízos da ordem de milhões de dólares ou reais. Logo,
  - É de extrema importância desenvolver métodos e meios para evitá-las/mitigá-las.
- Simulações de seu comportamento são importantes pois seus resultados em tempo de projeto permitem,
  - Antecipar potenciais problemas em tempo de voo.
- Seus subsistemas contribuem individualmente para o risco total da missão. Um dos mais influentes é o Sistema de Controle de Atitude (SCA). Consequentemente,
  - Simular este sistema e sua interação com a planta e ambiente espacial contribui para a análise e redução de riscos da missão.

1.2 Objetivo geral

Consideradas as motivações e justificativas, este trabalho:

Estuda a migração e a implementação de um software de controle de atitude para um sistema operacional e um processador em tempo real.

1.3 Objetivos específicos

- Recupera e preserva um ambiente e um software de SCA já desenvolvido por Amorim III (2009) para o sistema operacional em tempo real RTEMS, que executa sobre um simulador do processador ERC32 e se comunica com um computador PC que simula o satélite e o ambiente espacial ("Mundo Externo");
- Migra-o para um ambiente semelhante; mas, o implementa sobre um processador ERC32 real;
- Simula-o e compara-o com os desenvolvimentos anteriores nos mesmos casos e nas mesmas condições.

1.4 Metodologia

Adotaram-se os seguintes passos para alcançar os objetivos do trabalho:

I Revisar a literatura sobre SCAs, o SOTR RTEMS e o processador ERC32 simulados e tentativas de execução de SCAs em processadores reais;

II Definir uma arquitetura para o sistema SCA-Satélite-Ambiente Espacial;

III Estudar o funcionamento de um kit de avaliação do processador ERC 32 (TSC695) com o SOTR RTEMS;

IV Estabelecer a comunicação entre o kit de avaliação e um computador PC com SO Linux;

V Desenvolver um sistema cíclico de comunicação (envio e recebimento de dados por iterações) entre o kit e o computador PC com SO Linux;

VI Executar o modelo de simulação do satélite e ambiente espacial no computador PC;

VII Migrar, implementar e executar o modelo de SCA sobre o SOTR RTEMS no processador ERC32;

VIII Executar o sistema completo constituído de, SCA sobre o SOTR RTEMS no processador ERC32 e o modelo de simulação do satélite e do ambiente espacial no computador PC;

IX Simular e os comparar nos mesmos casos e nas mesmas condições dos desenvolvimentos anteriores.

2 CONCEITOS BÁSICOS E REVISÃO DA LITERATURA

2.1 Introdução

O material de estudo utilizado neste trabalho é composto por um livro sobre sistemas em tempo real, pela literatura aberta, na forma de documento técnicos, teses, dissertações e trabalhos de iniciação científica sobre simulação e/ou migração de SCA's em sistemas operacionais de tempo real.

2.2 Conceitos básicos

2.2.1 Sistema, modelagem e simulação

Este trabalho é fundamentado em modelagem e simulação; portanto, é necessária a definição de alguns conceitos principais e outros relativos a estes.

Sistema: é um conjunto de elementos, entidades ou partes que interagem entre si. Surgem das interações características que não apareceriam para cada elemento, entidade ou parte individualmente. A esta propriedade dá-se o nome de Propriedade Emergente dos sistemas ou simplesmente Emergência. Sistemas naturais apresentam características emergentes as quais são estudadas pelo ser humano, como por exemplo o Sistema Solar. Sistemas artificiais (feitos pelos seres humanos) também apresentarem características emergentes; porém, estes são designados para um determinado fim ou objetivo, como, por exemplo, um carro, que possui a finalidade de transportar pessoas ou objetos.

Modelagem: Segundo Souza e Trivelato (2003) modelagem é o ato de fazer uma descrição (das relações) de um sistema, processo, fenômeno, ser ou entidade em algum meio através de uma linguagem, com um objetivo ou ponto de vista baseado em leis da Natureza e medidas de seus parâmetros.

Modelo: Segundo Souza e Trivelato (2003) é a descrição feita pelo produto da modelagem. Este pode ser por exemplo: físico, lógico, matemático ou computacional.

Simulação: Segundo Souza e Trivelato (2003) é o ato de fazer uma descrição da evolução de um sistema, processo, fenômeno, ser ou entidade em um meio através de uma linguagem com um objetivo ou ponto de vista. A evolução é a mudança de estados (Valores das variáveis) durante um determinado

período de tempo. O interesse principal da simulação é o comportamento do sistema, seja apenas para análise ou para uso.

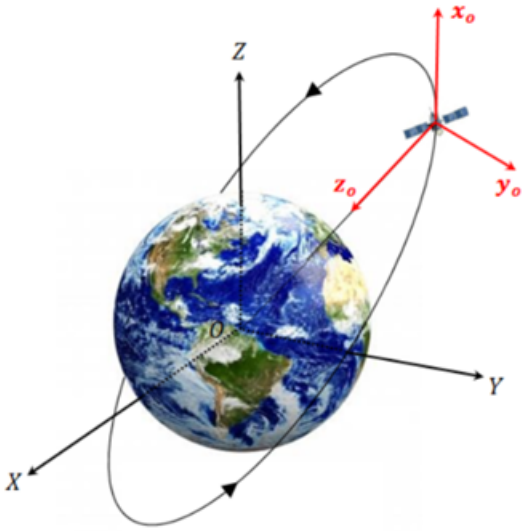
Simulador: Segundo Souza e Trivelato (2003) é o agente que realiza a simulação.

Fidelidade: É o nível que indica o quão bem um modelo ou simulador se adequa a realidade do objeto simulado. Segundo Amorim III (2009) deve se distinguir pelo menos dois tipos de fidelidade: fidelidade do modelo e fidelidade do simulador. A fidelidade do modelo depende da descrição do comportamento do sistema. A fidelidade do simulador depende dos métodos e meios utilizados para evoluir o modelo no tempo. À medida que se adicionam mais elementos ou componentes no modelo ou simulador sua fidelidade aumenta; porém, o simulador ou modelo se torna mais complexo (menos simples). Maior complexidade (ou menor simplicidade) implica mais dificuldade de implementação e análise do modelo ou simulador.

2.2.2 Software de controle de atitude e órbita

O Sistema de Controle de Atitude e Órbita pode ser dividido conceitualmente em duas partes principais. O controle de atitude que é responsável por controlar a posição e velocidade angulares de um satélite em torno de seu próprio centro de massa (movimento de rotação) o que é importante para o apontamento do satélite. O controle de órbita é responsável por controlar a posição e velocidade lineares do centro de massa de um satélite em relação a uma órbita desejada. Ambos utilizam um sistema de coordenadas: o controle de atitude utiliza um sistema de coordenadas com origem no centro de massa do próprio satélite (em vermelho na Figura 2.1). O controle de órbita utiliza um sistema de coordenadas com origem no centro da Terra (em preto na Figura 2.1).

Figura 2.1 - Modelo de Órbita e Atitude.



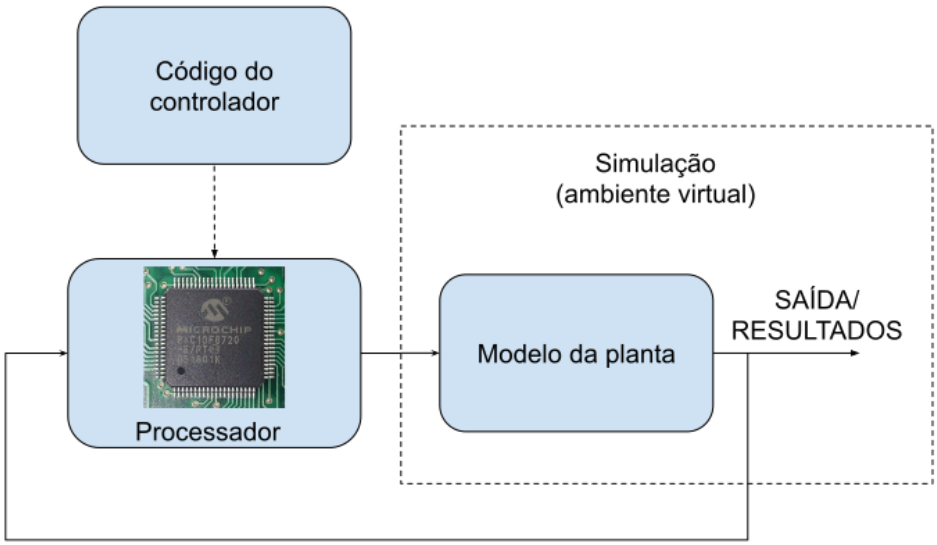
Fonte: Kuga et al. (2014).

O presente trabalho considera apenas o controle de atitude pois os modelos herdados do trabalho anterior contemplam apenas o controle de atitude. Portanto neste trabalho tratará apenas o Software de Controle de Atitude (SCA).

2.2.3 Processor-In-the-Loop (PIL)

É uma técnica de projeto que permite testar e verificar um código para sistemas embarcados diretamente em um hardware (alvo) dedicado (KALE et al., 2020). Um exemplo é um sistema de controle embarcado, em que o código a ser testado é o software de controle e o hardware dedicado pode ser um microcontrolador PIC ou DSP. Esta técnica permite a adequação do código às particularidades de um determinado hardware, tais como: a precisão da aritmética de ponto fixo, base de tempo (clock), limite de memória e modos diversos de otimização do compilador (KALE et al., 2020). Um diagrama da técnica pode ser visto na Figura 2.2:

Figura 2.2 - Diagrama PIL.



Fonte: O Autor.

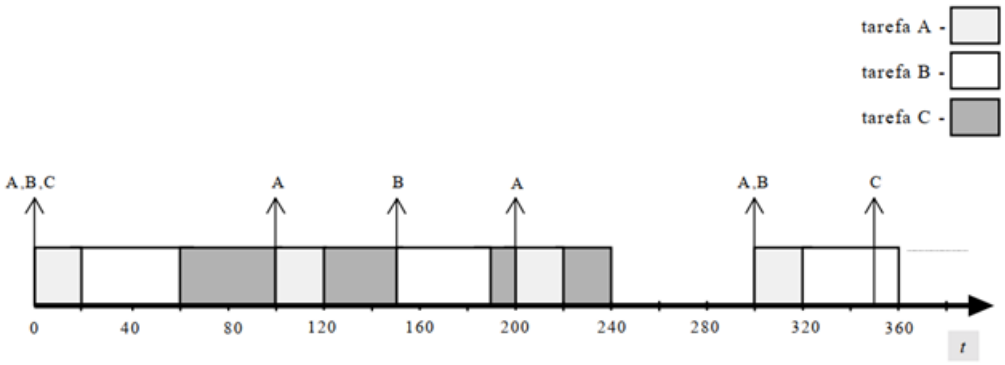
Esta técnica foi utilizada neste trabalho para testar o Software de Controle de Atitude.

2.2.4 Sistemas em tempo real

São sistemas de computação em que as tarefas realizadas devem ser logica e temporalmente corretas. Um sistema em tempo real deve possuir meios para verificar se os valores resultantes de qualquer operação estão logicamente corretos e para verificar se as tarefas ou operações obedecem às restrições temporais de projeto. Sistemas de computação regulares priorizam meios de correção lógica enquanto que, em sistemas em tempo real, meios de correção temporal são tão importantes quanto os de correção lógica. As restrições temporais de uma tarefa ou operação podem ser prazo, tempo de início, tempo de finalização, duração, etc., que estão diretamente relacionadas à preemptabilidade (capacidade de interrupção e retomada de uma tarefa) e previsibilidade. Segundo Stankovic (1988) a previsibilidade é a característica mais importante de um sistema em tempo real pois esta significa que o sistema deve possuir comportamento lógico e temporal tão determinístico quanto possível para atender aos requisitos de sistema. A característica de tempo real é relativa ao se considerar uma relação de sistemas cliente-servidor. Supondo que um cliente (aquele que requisita uma tarefa) demande uma tarefa e aguarde seus resultados com dadas restrições temporais; e, caso o sistema servidor (aquele que atende a tarefa) entregue

a tarefa com resultado correto e atenda às restrições temporais, então o sistema cliente percebe o sistema servidor como um sistema em tempo real. Em contrapartida, caso o sistema servidor viole as restrições temporais, o sistema cliente o perceberá como um sistema de tempo não-real. A Figura 2.3 demonstra o pedido de execução de tarefas (setas) e a execução das mesmas (barras com diferentes cores).

Figura 2.3 - Exemplos de tarefas distribuidas no tempo.



Fonte: Farines et al. (2000).

2.2.5 Sistemas operacionais em tempo real (SOTR)

São sistemas operacionais desenvolvidos para atender às características de tempo real dos softwares que serão executados sobre eles; em outros termos, impõem as restrições temporais citadas anteriormente sobre os softwares. As restrições temporais são impostas por um programa do SOTR chamado Agendador. Este pode ser implementado de várias formas; é sua devida configuração pelo projetista que determina o atendimento das restrições temporais. Para um SOTR, o agendador é em tempo de projeto, uma diretiva ou função configurável via programação por meio de uma interface de usuário. Por outro lado, para configurar um agendador sem um SOTR, o projetista deveria programá-lo diretamente na linguagem e no hardware de interesse, o que é mais complexo, pois demanda muito conhecimento específico da linguagem e do hardware. Consequentemente, isto requer mais tempo de projeto. Um SOTR facilita o processo de projeto de um sistema em tempo real, pois o problema do conhecimento específico é contornado pelos desenvolvedores do SOTR. Portanto, a grande vantagem de um SOTR é tornar o agendamento uma tarefa a nível de programação para o projetista. A desvantagem de um SOTR, é que o proje-

tista fica restrito ao uso das funções (diretivas) e bibliotecas fornecidas pelo sistema. Caso estas não atendam ao requisito esperado, o projetista pode necessitar realizar alterações nas mesmas sob pena de causar inconsistências no sistema operacional. Essas inconsistências podem causar, por exemplo, perda de desempenho ou violar regiões de memória do sistema, o que acarretaria em uma provável falha ou falência do mesmo. A vantagem de um projeto direto do agendador, é que o projetista pode utilizar diretamente as instruções da plataforma alvo, isto permite que o mesmo possa otimizar o desempenho do agendador.

Os SOTR podem ser subdivididos em críticos e não-críticos.

Críticos (Garantia prevista): São sistemas que garantem, em tempo de projeto, que as restrições temporais das tarefas sejam atendidas em tempo de execução. Estes sistemas não toleram desvios temporais. Um exemplo são os sistemas de controle de uma aeronave, em que o não cumprimento de uma restrição temporal pode significar a perda de vidas humanas ou grande prejuízo material. O Sistema VxWorks da WindRiver é um exemplo de SOTR crítico. Dois casos de uso são os rovers de exploração em Marte: Spirit e Opportunity (WINDRIVER, 2020).

Não-Críticos (Melhor esforço): São sistemas que não garantem o cumprimento das restrições temporais das tarefas executadas porém realizam o maior esforço possível para cumprí-las. Estes sistemas toleram eventuais não cumprimentos de restrições temporais. Um exemplo são os sistemas de mídia que executam músicas e vídeos: um atraso na execução é incômodo, porém não traz grande prejuízo material ou ameaça a vida humana.

2.2.6 Real-time Executive For Multiprocessor Systems (RTEMS)

Segundo RTEMS (2020), o Real-Time Executive for Multiprocessor System (RTEMS) é um sistema operacional de tempo real. Possui características tais como:

- Não há separação entre espaço de usuário e espaço de núcleo (kernel);
- Multi-threading;
- Multi-processamento simétrico (MPS);
- Suporte para diferentes arquiteturas (Arm, Microblaze, Sparc, etc...);

- Código fonte aberto;
- Agendadores configuráveis (prioridade fixa em nível de tarefa e sub-tarefas);
- Agendamento em cluster (quando em MPS);
- Possui um ecossistema de desenvolvimento (facilita adequação/personalização).

O RTEMS possui diretivas (funções) que permite o projetista não ter necessidade de diretamente controlar ou sincronizar diferentes tarefas e processadores. É usado em aplicações espaciais, médicas, redes e outras aplicações de sistemas embarcados (RTEMS, 2020).

Este trabalho utiliza o RTEMS como SOTR pois herdou simuladores diretamente do trabalho de Amorim III (2009) que os herdou de trabalhos anteriores. A versão utilizada neste trabalho é a 4.10 pois foi fornecida e exaustivamente executada pela Divisão de Eletrônica Aeroespacial (DEA) do INPE. É importante notar que a DEA realizou uma extensa pesquisa para adotar um SOTR de base para a PMM, instanciada no satélite Amazonia-1. O RTEMS foi escolhido entre outras devido às seguintes motivações:

- É Certificado pela ESA (ESA, 2021);
- Oferece suporte para o processador ERC 32 disponível na época;
- Tem código fonte aberto;
- É adotado em missões espaciais e militares.

2.2.7 Processador ERC32

Segundo o documento Saab (1997), o projeto do processador ERC32 (Embedded Real-time Computing core - 32 bits) foi demandado pela ESA (European Space Agency) com o objetivo de desenvolver um hardware capaz de realizar operações com 32 bits em tempo real. As características principais do hardware foram focadas em desempenho, deteccção de erro simultânea, capacidade de manuseio, resistência à radiação e consumo de energia. Os chips da época eram limitados a 16 bits e os novos projetos espaciais demandavam maior capacidade de processamento. Para reduzir o custo foi requerido que a arquitetura do hardware fosse baseada em alguma

já existente. A arquitetura escolhida foi a SPARC (Scalable Processor Architecture). O projeto foi coordenado pela Saab Ericsson que designou outras empresas para as diferentes partes e etapas do mesmo. O ERC32 consistia de três circuitos integrados (chipset): Unidade de Ponto Flutuante (UPF), Unidade de Inteiros (UI) e Controlador de Memória(CM). As características atingidas pelo chip na época foram:

- Desempenho: Até 10 milhões de instruções por segundo em 14 MHz de frequência (clock);
- Detecção de erro: Mais de 95% de detecção de erro (memória inclusa);
- Resistência à radiação: Imune às correntes parasitas para até 70kRad;
- Consumo de energia: Menor que 2,25 W a 5,5 V.

A empresa Temic foi responsável pela manufatura do ERC32 na época. Em 1998, a Atmel adquiriu a Temic e o processador passou a ser constituído de um único chip que contéma a UPF, UI e o CM integrados. O novo chip recebeu o nome de TSC695. Em 2016, a Microchip adquiriu a Atmel e ainda mantém em seu site toda a documentação sobre o TSC695. Este trabalho foi desenvolvido sobre um kit de avaliação do processador ERC 32 (TSC695) com o SOTR RTEMS.

A ESA continuou o estudo para desenvolver um sucessor do ERC32. O resultado foi a família de processadores LEON baseado na arquitetura SPARCV8. Os modelos LEON1 e LEON2 foram desenvolvidos pela ESA. A Gaisler Research, desenvolveu a partir do modelo LEON3. Atualmente a empresa, parte da Cobham, desenvolve duas famílias de processadores: a LEON SPARC V8 e a NOEL-V RISC-V. O último processador desenvolvido para família LEON é o LEON5 (GAISLER, 2021).

2.2.8 Plataforma MultiMissão (PMM)

A Plataforma MultiMissão é uma plataforma genérica (Módulo de Serviço) para satélites de até 500 Kg com cargas úteis de aproximadamente 280 Kg. O propósito é prover todos os equipamentos necessários para o funcionamento básico de um satélite independente do tipo de órbita. A vantagem do conceito de desenvolver uma plataforma genérica é que após o desenvolvimento e qualificação da primeira plataforma, sua reprodução dispensará demoradas e custosas etapas de desenvolvimento e qualificação que são necessárias em um projeto novo. Desta forma, a equipe de desenvolvimento necessitaria de orientar seus esforços para adequar a plataforma à uma missão diferente e não precisaria de criar e desenvolver um projeto novo

para cada nova missão. Consequentemente, isto reduziria custos e prazos de missões subsequentes (INPE, 2020).

As funcionalidades básicas fornecidas pela PMM são:

- Apontamento;
- Geração de energia;
- Controle térmico;
- Gerenciamento de dados;
- Telecomunicação de serviço.

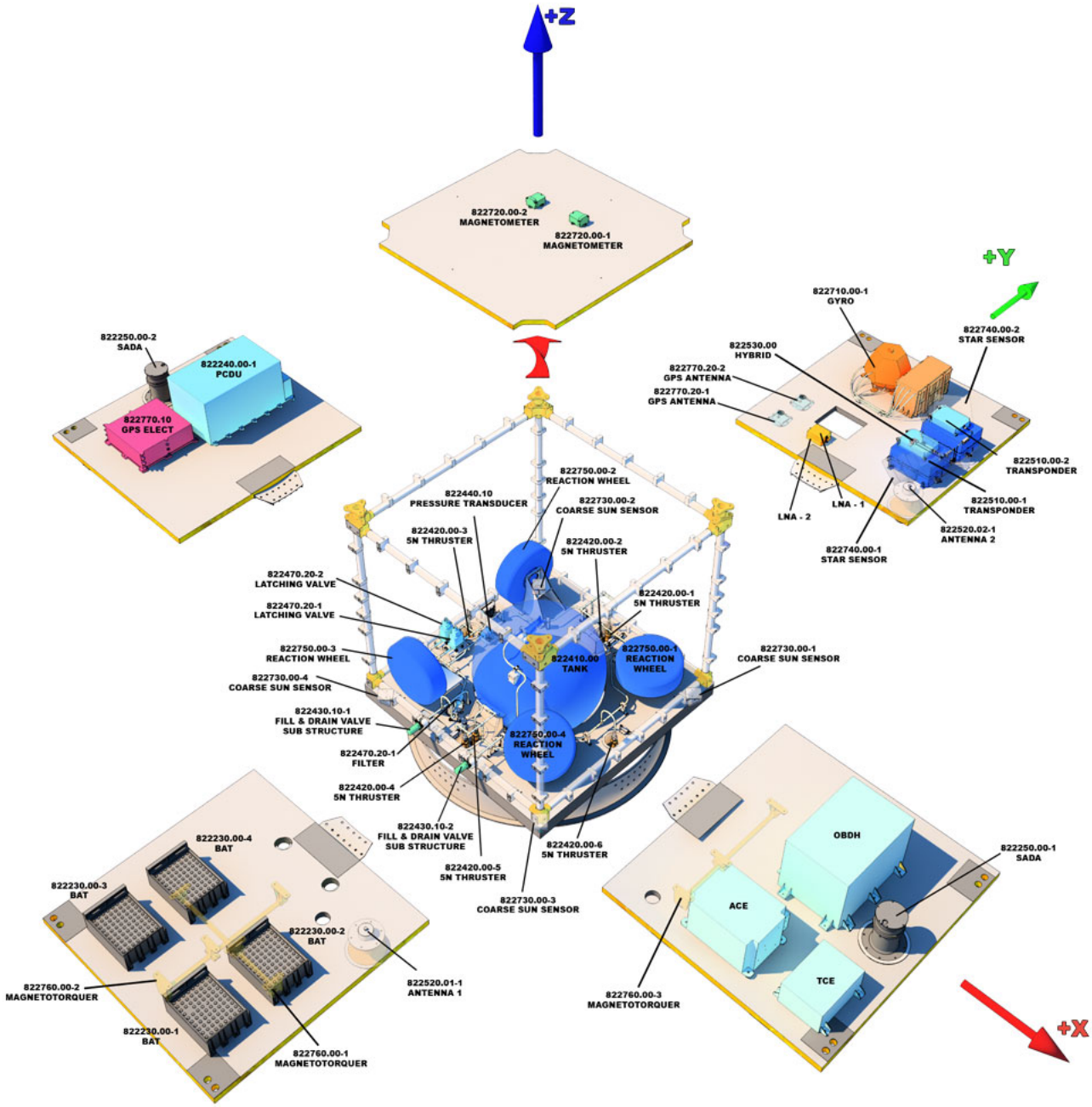
A PMM é composta por vários subsistemas, cada um responsável por uma função; estes são:

- Subsistema de Estrutura Mecânica do Módulo de Serviço;
- Subsistema de Controle de Atitude e Órbita e Supervisão de Bordo;
- Subsistema de Propulsão;
- Subsistema de Telemetria e Telecomando;
- Subsistema de Suprimento de Energia (inclui Gerador Solar).

Atualmente, a PMM está em órbita a serviço do satélite Amazonia 1. A missão deste é a Missão Amazônia, que tem por objetivo fornecer dados de sensoriamento remoto para observação e monitoramento de desmatamento, especialmente na região amazônica.

A Figura 2.4 é uma vista esquemática aberta da atual versão da PMM.

Figura 2.4 - Vista esquemática aberta da PMM.



Fonte: INPE (2020).

Este trabalho utiliza modelos similares aos da PMM de 2001 utilizados no trabalho de Amorim III (2009) por ser continuação direta do mesmo. Portanto, os modelos utilizados descrevem: a dinâmica do satélite, o ambiente espacial e a parte de atitude

do Subsistema de Controle de Atitude e Órbita e Supervisão de Bordo.

Segundo Amorim III (2009) a PMM de 2001 deve se manter dentro das seguintes especificações:

- Apontamento: $< 0,05^o$ (3σ)
- Deriva: $< 0,001^o/\text{s}$
- Determinação de atitude: $\leq 0,005^o$ (3σ)
- Reposicionamento de até 30^o em 180 segundos

2.3 Revisão da literatura

O conhecimento desenvolvido neste trabalho é baseado em livros, artigos e dissertações e trabalhos de iniciação científica anteriores de mesma natureza.

Os conceitos de tempo real foram baseados no artigo de Stankovic (1988) e no livro Sistemas de Tempo Real (FARINES et al., 2000). O conceito de simulação foi visto em Souza e Trivelato (2003)

Este trabalho é a continuação de uma sequência de trabalhos sobre o desenvolvimento de SCA's para operar em tempo real, a saber:

- Gilberto da Cunha Trivelato (TRIVELATO, 1988)
  Desenvolveu um sistema de controle em tempo real de um protótipo de uma roda de reação por meio de um controlador digital em um protótipo de computador digital ASTROS LV2.
- Silvano Vargas Prudêncio (PRUDENCIO, 2000)
  Simulou em tempo real um sistema de controle de atitude por meio de bobinas magnéticas baseado nos parâmetros do satélite SACI-1. Porém, em seu trabalho, a simulação foi realizada apenas em um computador com e sem a geração do código C do controlador no ambiente MatrixX. Comparou os valores obtidos pelos dois modos do modelo do satélite, dos sensores e atuadores.
- Domingos Savio Barbosa (BARBOSA, 2000)
  Simulou em tempo real de um sistema de controle de atitude por meio

de bobinas magnéticas baseado nos parâmetros do satélite SACI-1. O simulador foi dividido em duas grandes partes, uma parte responsável pelo controlador que executa em um PC (386) e outra parte responsável por simular a dinâmica do satélite, sensores, atuadores, campo magnético e sol em um outro PC (386). A malha é fechada por meio de uma comunicação serial entre os dois computadores via 2 placas Megatel com 8 portas seriais.

- Marcelo Ricardo Alves da Costa Tredinnick(TREDINNICK, 1999a)
  Estudou e simulou um controle discreto de atitude de satélites artificiais com apêndices flexíveis e analisou a estabilidade dos mesmos em função de períodos de amostragem crescentes.

- Patrícia Trindade Araújo (TREDINNICK, 1999b)
  Estendeu o trabalho de Prudêncio por meio da adição de um simulador dinâmico físico de 3 eixos na malha de controle. O mesmo foi responsável por imitar a dinâmica do satélite SACI-1.

- Marcio Ferraz Gobato(GOBATO, 2006)
  Estudou e comparou dois métodos de projeto do Controle Clássico (Uma Entrada, Uma Saída) com três métodos de projeto do Controle Moderno (Múltiplas Entradas, Múltiplas Saídas) aplicados a sistemas aeroespaciais fracamente ou fortemente acoplados. Teve como primeira aplicação modelos similares aos da PMM de 2001, então em desenvolvimento no INPE.

- Marcelo de Lima Bastos Moreira (MOREIRA, 2006)
  Desenvolveu uma simulação em tempo real de um sistema de controle de atitude por meio de rodas de reação baseado nos parâmetros de modelos similares aos da PMM de 2001. O simulador foi dividido em duas grandes partes, uma parte foi responsável pelo controlador, atuadores, dinâmica do satélite e sensores e a outra parte foi responsável por simular o ambiente espacial. A primeira parte foi executada em um simulador de instrução SPARC, afim de se testar o código na arquitetura do processador ERC 32. Este foi escolhido como processador da PMM. Tal dissertação é uma das bases deste trabalho e, por isto, é detalhada no capítulo 3.

- Francisco Carlos de Amorim III (AMORIM III, 2009)
  Continuou o trabalho de Moreira (2006) porém reorganizou a estrutura do simulador e realizou as simulações em cinco implementações diferentes. Tal dissertação é uma das bases deste trabalho e, por isto, é detalhada no

capítulo 3. Quatro implementações são relevantes para este trabalho e, por isto, são detalhados na Seção 4.1.

- Denis Francisco Simões Donizete (DONIZETE et al., 2011)
  Fez um trabalho de iniciação científica baseado no trabalho de Amorim III (2009) em que propôs implementar o software de controle de atitude em um processador ERC32 real situado em um kit VME SPARC da Tharsys. Avançou parcialmente devido às dificuldades técnicas.

- João Marcos Alves Ballio Barreto (BARRETO et al., 2012)
  Continuou diretamente o trabalho de iniciação científica feito por Donizete et al. (2011), cuja proposta foi preservada até onde avançou; mas não obteve êxito devido à complexidade do problema e incompatibilidade de sistema.

- Gitsuzo de Brito Siqueira Tagawa (TAGAWA, 2013)
  Modelou, implementou e simulou o modo nominal de um SCA para modelos similares aos da PMM de 2001 em um Simulador de IMA (SIMA).

- André Luiz Oliveira Moraes (MORAES, 2017)
  Continuou o trabalho de Tagawa (2013). Apreciou, simulou e implementou a arquitetura Aviônica Modular Integrada e Distribuída (Distributed Integrated Modular Avionics - DIMA) e a aplicou a um SCA para modelos similares aos da PMM de 2001.

- Palmira dos Santos (SANTOS, 2020)
  Continuou o trabalho de Moraes (2017). Usou um SCA como caso de uso para avaliação de um concentrador de dados remoto em uma arquitetura DIMA (Distributed Integrated Modular Avionics) aplicado ao modo normal de um SCA para modelos similares aos da PMM de 2001.

  O presente trabalho é uma continuação direta do trabalho de Amorim III (2009), e, assim como Donizete et al. (2011) e Barreto et al. (2012), se propôs a implementar o software de SCA em um processador ERC32 real situado em um kit de desenvolvimento.

3 MODELOS

3.1 Escolha da missão

Os estudos deste trabalho são sobretudo baseados nos trabalhos de Gobato (2006), Moreira (2006) e Amorim III (2009). Nos momentos em que os mesmos foram realizados, não havia uma missão específica definida para a PMM, mas havia os documentos e os modelos de 2001. Como este trabalho herdou os estudos e abordagens dos trabalhos anteriores, os modelos e seus parâmetros são os mesmos (de 2001), diferem apenas na implementação. Entretanto, na época já havia o interesse em realizar uma missão de monitoramento da Floresta Amazônica Brasileira e esta já era uma possível candidata para primeiro uso da PMM.

Segundo o INPE (2021), no presente momento, a Missão Amazônia está em andamento e é a primeira a ser realizada por meio da PMM. O primeiro satélite destinado à esta missão é o Amazonia 1 que já se encontra em órbita, visando inaugurar e validar a arquitetura da PMM (INPE, 2020).

A Missão Amazônia tem como objetivo fornecer imagens para observar e monitorar o desmatamento, especialmente na região amazônica e também as atividades agrícolas do país. Grande parte da vegetação brasileira está sob forte atividade antropogênica tais como extração de madeira, agricultura, manejo de pastagens etc. Portanto, é importante observar como estas atividades progridem em relação aos biomas.

Há fatores importantes na agricultura brasileira que influenciam a qualidade da observação do satélite. Alguns destes são a dimensão da cultura, o período entre a semeadura e a colheita e o calendário agrícola. A dimensão da cultura, varia de pequenas lavouras a culturas de grande porte. Este fator aliado a fenômenos de desmatamento em grande escala requer uma alta resolução espacial das imagens obtidas.

O calendário agrícola brasileiro coincide com os períodos de chuvas; portanto, há alta probabilidade de haver cobertura de nuvens. O período entre a semeadura e a colheita de certas culturas, como por exemplo: milho, soja é curto. Estes fatores contribuem para uma menor probabilidade de se obter um dado óptico útil acerca do progesso das culturas. Portanto, isto requer um período curto de revisita do satélite sobre as regiões de interesse, a fim de se aumentar a probabilidade de se obter um dado útil. A Missão Amazônia considera estes fatores e portanto o satélite Amazonia 1 é projetado para atender estes e outros fatores não citados (INPE, 2020).

A validação de conceito da PMM permite que a mesma arquitetura seja reutilizada em projetos de satélites subsequentes.

Os parâmetros orbitais utilizados neste trabalho serão os mesmos do trabalho de Amorim III (2009), pois os modelos utilizados foram herdados de seu trabalho. Estes são:

- Altitude: 905 km
- Inclinação: 0°
- Excentricidade: 0 (circular)

3.2 Herança e histórico dos modelos

Os modelos similares aos da PMM de 2001 utilizados nesta dissertação foram herdados diretamente do trabalho de Amorim III (2009). O trabalho deste herdou os modelos que foram desenvolvidos por Gobato (2006) e Moreira (2006). Estes por sua vez foram baseados nos modelos para o satélite SACI-1 desenvolvidos por Prudencio (2000) e Barbosa (2000), todos no ambiente de modelagem MatrixX/SystemBuild.

Gobato (2006) e Moreira (2006) adaptaram os modelos do satélite SACI-1 feitos por (PRUDENCIO, 2000) e Barbosa (2000) para modelos similares aos da PMM de 2001, pois esta possuia características diferentes das do satélite SACI-1. Os parâmetros físicos não são os mesmos e a PMM de 2001 possui rodas de reação para sua estabilização.

O objetivo da dissertação de Amorim III (2009) foi de aumentar o grau de fidelidade das simulações dos modelos similares aos da PMM de 2001 e não dos modelos. Portanto, os modelos utilizados por Amorim III (2009) são exatamente os mesmos de Moreira (2006), porém diferem nos ambientes e configuração de simulação.

Amorim III (2009) realizou cinco implementações em ambientes computacionais distintos. Estas são detalhadas na Seção 4.1.

O presente trabalho herda os modelos utilizados por Amorim III (2009); portanto os modelos são os mesmo utilizados por Gobato (2006) e Moreira (2006), mas diferem apenas em implementação. Os detalhes desta estão no próximo capítulo.

3.3 Ambiente de modelagem

3.3.1 Suíte MatrixX

O ambiente utilizado para o desenvolvimento de um SCA para modelos similares aos da PMM de 2001 foi a suíte de programas MatrixX da National Instruments. Esta é formada por cinco componentes Xmath, SystemBuild, AutoCode, DocumentIt e RealSim. Neste trabalho são usados sobretudo o SystemBuild e o AutoCode. O objetivo da suíte é assistir no desenvolvimento no sistema de controle embarcados em tempo real.

Gobato (2006) e Moreira (2006) desenvolveram os modelos da PMM de 2001 no componente SystemBuild em forma de diagrama de blocos. A modelagem por diagrama de blocos permite agrupar partes específicas do sistema e formar sub-sistemas. A vantagem desta abordagem, é que é possível verificar parte dos sistemas incrementalmente.

A National Instruments informou em Junho de 2018 em seu fórum que o MatrixX entrou em seu ciclo de fim de vida (NI FORUM, 2020). O suporte do mesmo acabará em Junho de 2023.

3.3.2 Geração automática de código

O Componente AutoCode é capaz de gerar códigos para sistemas em tempo real automaticamente nas linguagens C ANSI ou ADA 95 a partir dos diagramas de blocos feitos no SystemBuild. Os códigos resultantes podem ser compilados para realizar simulações ou serem adequados aos alvos (hardware ou plataformas) específicos com a finalidade de desenvolver sistemas embarcados. O procedimento para geração de código pode ser visto na Figura 3.1. Os passos do procedimento estão descritos detalhadamente em NI (2004) e de maneira resumida são:

I Construir um modelo em diagrama de blocos do sistema por meio do SystemBuild e verificá-lo mediante simulação.

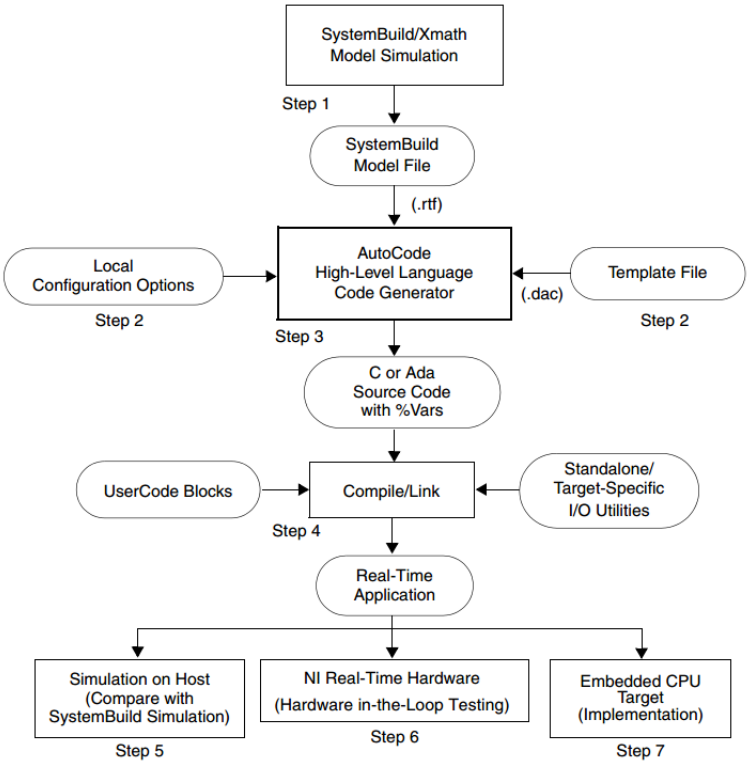
II Personalizar a geração de códigos por meio das opções de configuração do ambiente MatrixX e por meio de gabaritos (*templates*). Segundo Amorim III (2009), isto permite sua adequação a um sistema operacional ou linguagem e hardware específico ou alvo.

III Gerar o código automático por meio da ferramenta AutoCode.

IV Compilar e ligar os códigos gerados. Nesta fase, é possível editar arquivos de entrada e saída gerados pelo AutoCode para adequá-los ao ambiente de execução do código. Além disso, é possível também adicionar blocos de códigos desenvolvidos pelo usuário a fim de se melhorar a funcionalidade do modelo.

Os passos V, VI e VII são destinados à simulação, teste e implementaação do código no sistema operacional ou linguagem e hardware específico ou alvo. Estes não foram detalhados pois não são relativos à geração do código.

Figura 3.1 - Procedimento para geração de código do MatrixX/AutoCode.



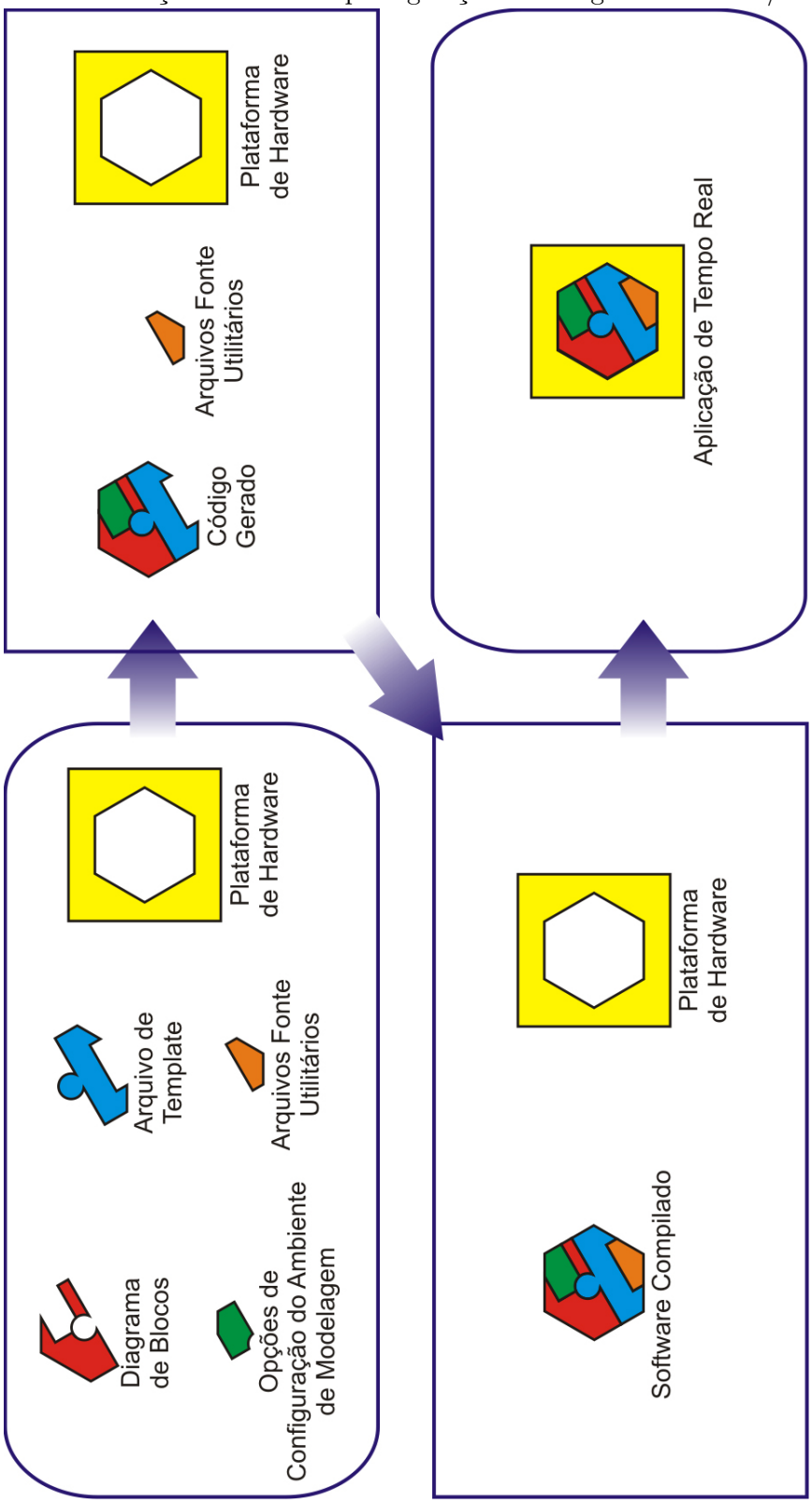
Fonte: NI (2004).

3.3.2.1 Personalização

O código gerado pode ser personalizado por meio de um gabarito (*template*), configurações locais, blocos de códigos programados pelo usuário e códigos específicos do hardware alvo, como, por exemplo, uma biblioteca para entrada e saída. Os dois

primeiros meios são configurações pré-geração de código, enquanto os dois últimos são pós-geração de código. As informações suficientes para geração de um código podem ser vistas na Figura 3.2.

Figura 3.2 - Informações suficientes para geração de código do MatrixX/AutoCode.

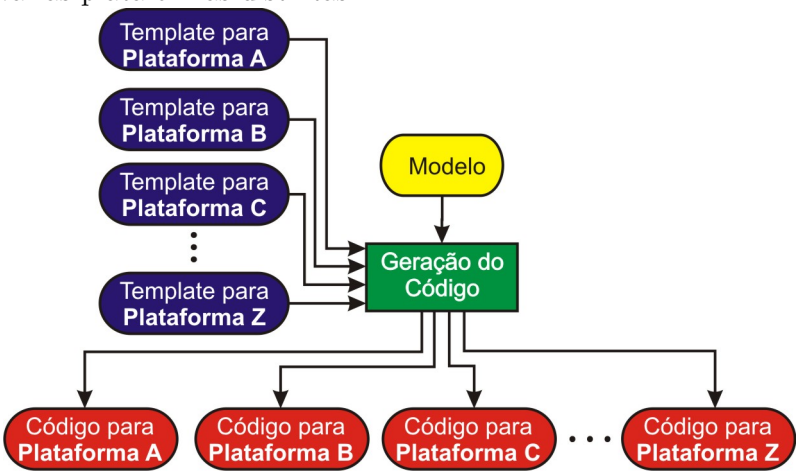


Fonte: Amorim III (2009).

Os gabaritos (*templates*) são particularmente úteis pois estes permitem uma padronização múltipla da arquitetura do código; consequentemente, isto confere versatilidade à adequação do modelo do SystemBuild a diferentes plataformas de destino/alvo. Segundo Amorim III (2009), é possível usar um mesmo modelo do SystemBuild para várias plataformas de destino/alvo ou usar vários modelos para uma mesma plataforma. Isto pode ser melhor compreendido nas Figuras 3.3 e 3.4.

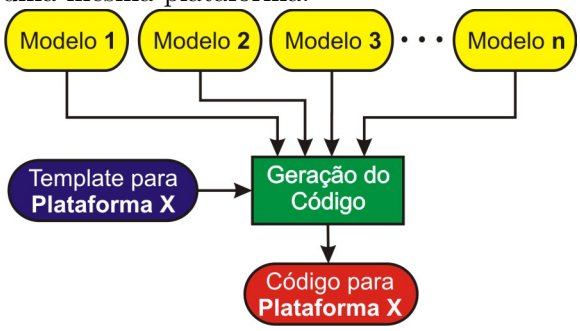
Os procedimentos para gerar um novo gabarito (*template*) podem ser vistos no fluxograma da Figura 3.5.

Figura 3.3 - Versatilidade do uso de *templates* para gerar código do mesmo modelo para várias plataformas distintas.



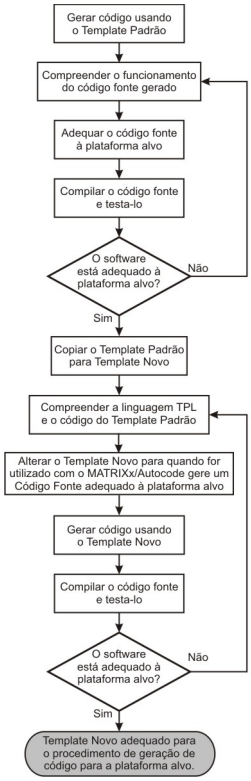
Fonte: Amorim III (2009).

Figura 3.4 - Versatilidade do uso de *templates* para gerar código de vários modelos distintos para uma mesma plataforma.



Fonte: Amorim III (2009).

Figura 3.5 - Procedimento para criação de um novo gabarito (*template*).



Fonte: Amorim III (2009).

3.3.2.2 Componentes do código gerado

Caso o usuário não realize modificações no gabarito (*template*) original, a ferramenta AutoCode irá gerar um código que consiste em agendador/gerenciador de aplicações de tempo crítico, despachador re-entrante, um ou mais subsistemas preemptíveis, funções de entrada e saída, manipulador de interrupção por tempo e uma função de plano de fundo. As chamadas de cada módulo gerado estão no gabarito (*template*). Isto permite ao desenvolvedor alterar a estrutura do código mediante a necessidade. Os componentes padrões gerado pelo gabarito (*template*) original são:

Agendador: É uma rotina de tempo crítico que realiza operações de entrada e saída para a aplicação gerada, gerencia tarefas de manutenção interna do software e gera uma lista de despacho de subsistemas que estão prontos para serem executados.

Despachador: Rotina que despacha os subsistemas que estão prontos para serem executados na lista de despacho gerada pelo agendador. Os sistemas de maior prioridade são despachados primeiro.

Subsistemas: Contém os códigos que implementam a lógica de funcionamento dos blocos do modelo desenvolvido no SystemBuild. Implementam as atividades em tempo real ao aceitar as entradas e disponibilizar as saídas em períodos de amostragem dos blocos maiores (grau de abstração superior). O período de amostragem é administrado pelo agendador.

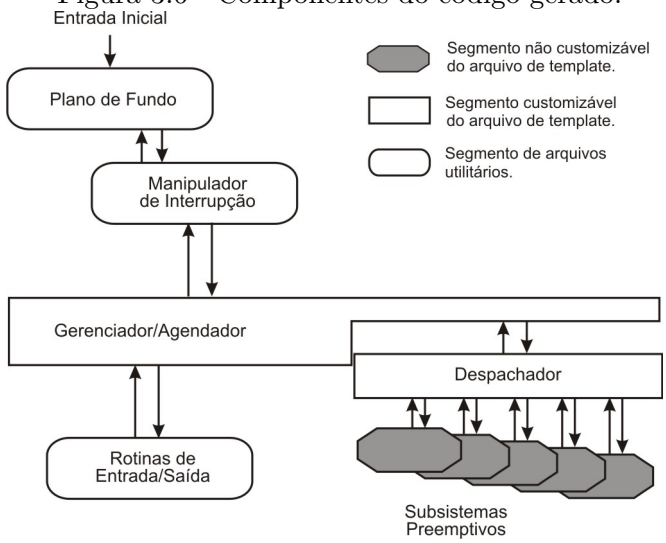
Rotinas de E/S: São rotinas que são responsáveis por providenciar entradas de dados para a aplicação em tempo real e obter as saídas da mesma.

Manipulador de interrupção por tempo: Rotina responsável por invocar o agendador em um intervalo de tempo específico.

Função de plano de fundo: Função responsável por realizar tarefas não temporalmente críticas tais como: auto-diagnose ou atualizar um display. Sua principal qualidade é ser interruptível.

Os componentes e suas relações podem ser melhor compreendidos na Figura 3.6:

Figura 3.6 - Componentes do código gerado.



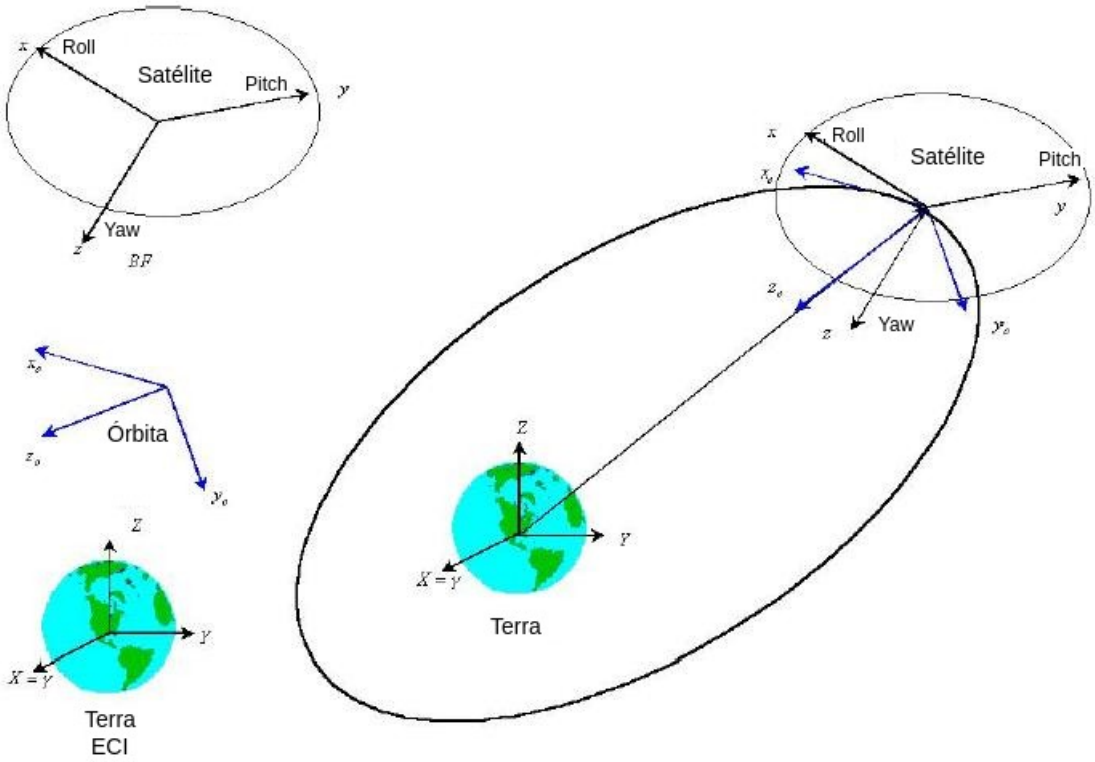
Fonte: NI (2004).

3.4 Sistemas de referência

Os sistemas de referência adotados neste trabalho são os mesmos adotados por Gobato (2006), Moreira (2006) e Amorim III (2009). Os três tipos básicos de sistemas de referência para descrever os movimentos do veículo espacial são: o fixo em relação ao corpo do veículo espacial; o fixo no "espaço inercial"; e o definido em relação à órbita e não fixo em relação ao veículo nem ao espaço inercial (WERTZ, 1978).

Os sistemas de referência são apresentados na Figura 3.7:

Figura 3.7 - Sistemas de referência.



Fonte: Adaptada de Arantes Júnior (2005).

3.4.1 Referencial inercial

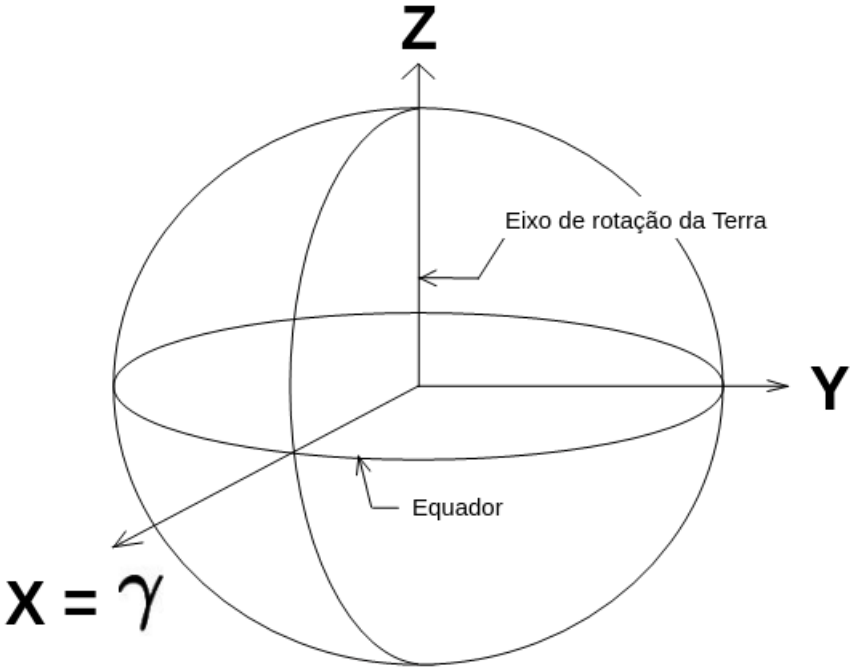
O sistema de referência fundamental, ao qual todo o movimento deve ser referenciado é o Referencial Inercial. Este, idealmente, seria um referencial absoluto imóvel em relação às estrelas "fixas", porém, em situações práticas, isto não é possível. Logo, adota-se uma constelação distante, de forma que o movimento relativo entre os demais sistemas seja insignificante. Para a maioria dos problemas, é suficiente escolher um sistema de referência cuja aceleração não seja alta a ponto de perturbar a solução do problema além da precisão requerida (KAPLAN, 1976). É dito "com direções inerciais".

O sistema de coordenadas "com direções inerciais"mais comum é o sistema de coordenadas celestes cuja origem é o centro de massa da Terra, cujo eixo Z é o eixo de rotação da Terra (WERTZ, 1978). O Pólo Norte Celeste está aproximadamente

a 1 grau da estrela Polaris. O ponto no Equador Celeste escolhido como ponto de referência para o eixo X é o ponto onde a Eclíptica, ou plano da órbita do Sol em torno da Terra, cruza o Equador indo do sul para o norte, conhecido como Equinócio Vernal. Essa é a direção da linha do centro da Terra para o Sol no primeiro dia de Primavera no Hemisfério Norte. Este sistema de coordenadas celeste não é verdadeiramente inercial; porém, é suficiente escolher um sistema de coordenadas que garanta a precisão requerida.

O presente trabalho adota o mesmo sistema de coordenadas adotado por Amorim III (2009), pois os modelos são os mesmos. O sistema adotado é o sistema de coordenadas celestes (X, Y, Z), que tem origem no centro da Terra. O eixo X aponta na direção do Ponto Vernal γ, o eixo Z aponta para a direção do Pólo Norte geográfico e o eixo Y completa o sistema dextrógiro. O sistema é mostrado na Figura 3.8:

Figura 3.8 - Sistema de referência "com direções inerciais"centrado na Terra.

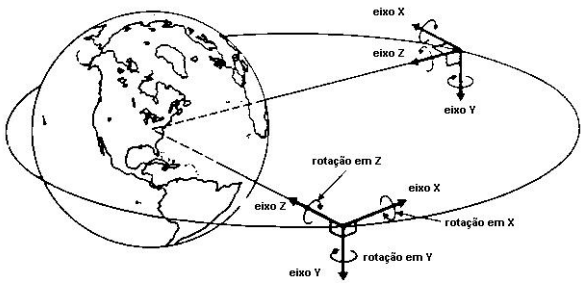


Fonte: Adaptada de Popescu (2021).

3.4.2 Referencial orbital

O Referencial Orbital adotado no presente trabalho é o mesmo selecionado por Amorim III (2009). Este é o sistema Vertical Local Horizontal Local (VLHL), (x_o, y_o, z_o). Nesse sistema, o eixo z_o é direcionado para o nadir (para o centro da Terra), o eixo y_o é direcionado para o negativo da normal à órbita, e o eixo de x_o é perpendicular aos outros dois, formando o sistema dextrógiro. O eixo x_o coincide com a direção do vetor velocidade orbital para o caso de uma órbita circular (WERTZ, 1978). O sistema é mostrado na Figura 3.9:

Figura 3.9 - Sistema de referência Vertical Local Horizontal Local.



Fonte: Adaptada de Wertz (1978).

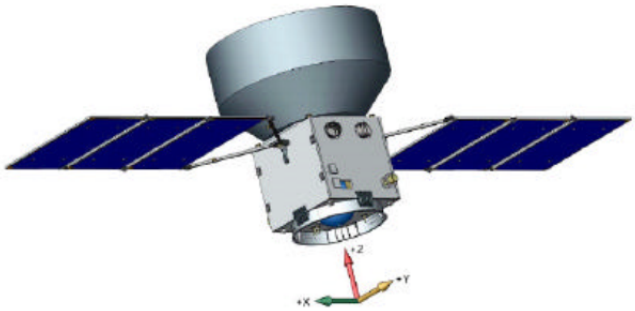
3.4.3 Referencial do satélite

O sistema de coordenadas fixo no satélite, Referencial do Satélite, é utilizado para definir a orientação para o sistema de determinação e controle de atitude. Este também é utilizado como referência para a medição da atitude.

Os eixos têm origem no centro de massa do satélite e são definidos como, rolamento (*roll*), arfagem (*pitch*) e guinada (*yaw*), que coincidem com os eixos dos momentos principais de inércia do satélite (x, y, z). Os ângulos de *roll*, *pitch* e *yaw* (ψ, θ, ϕ) são definidos como rotações em relação aos respectivos eixos, e obedecem à regra da mão direita.

Esse sistema é o mais utilizado para satélites orientados para a Terra (WERTZ, 1978). O sistema de referência do corpo é ilustrado pela Figura 3.10:

Figura 3.10 - Sistema de referência no corpo.



Fonte: Gobato (2006).

3.5 Modelo da planta

A seguinte notação foi convencionada para os sistemas de referência utilizados para modelagem da cinemática e dinâmica da planta:

- Referencial Inercial: N
- Referencial Orbital: A
- Referencial do corpo do satélite: B

De acordo com Gobato (2006) e Moreira (2006), algumas simplificações foram consideradas para o modelo similar ao da PMM de 2001, entre elas:

- satélite como um corpo rígido
- painéis solares fixos ao satélite

3.5.1 Dinâmica

A equação diferencial do movimento rotacional de um corpo rígido a um referencial inercial (N) mas reescrita no referencial do corpo (B) é dada por:

$$\dot{\vec{h}} + \vec{w}_{B/N} \times \vec{h} = \vec{T}_p \tag{3.1}$$

como momento angular é $\vec{h} = I\vec{w}$ a equação fica:

$$I\dot{\vec{w}}_{B/N} + \vec{w}_{B/N} \times (I\vec{w}_{B/N}) = \vec{T_p} \tag{3.2}$$

em que $\vec{w}_{B/N}$ é o vetor velocidade angular do sistema de referência do corpo (B) em relação a um sistema inercial de coordenadas (N), I é a matriz de inércia do corpo e $\vec{T_p}$ é o torque devido às perturbações externas.

Para um veículo espacial rígido (WIE, 1998) em que se considera a presença de rodas de reação e seus acoplamentos de inércia com o satélite (WERTZ, 1978) a equação fica:

$$I\dot{\vec{w}}_{B/N} + \vec{w} \times (I\vec{w}_{B/N}) = -\vec{w}_{B/N} \times h_r - \dot{h}_r + \vec{T_p} \tag{3.3}$$

em que I é a matriz de inércia do satélite, $\vec{w}_{B/N}$ é o vetor velocidade angular do satélite (referencial do satélite B em relação ao referencial inercial N), h_r é o momento angular das rodas de reação, $\dot{h}_r$ é a variação do momento angular das rodas de reação, que representa a parcela controlável da equação, também chamado de torque de controle e $\vec{T_p}$ é o torque devido às perturbações externas.

3.5.2 Cinemática

A atitude pode ser representada por ângulos de Euler e/ou quaternions. No trabalho de Amorim III (2009) apenas a representação por ângulos de Euler foi utilizada. As equações de cinemática a seguir permitem simular o comportamento da atitude do satélite.

Considere um referencial orbital chamado de A (MOREIRA, 2006), que apresenta origem no centro de massa do satélite e tem vetores unitários $(\vec{a}_1, \vec{a}_2, \vec{a}_3)$, sendo: $\vec{a}_1$ na direção da órbita, $\vec{a}_2$ perpendicular ao plano da órbita e $\vec{a}_3$ em direção à Terra (WIE, 1998) como ilustrado na Figura 3.9. A velocidade de A para uma órbita circular em relação a um referencial inercial, chamado de N, é:

$$\vec{w}_{A/N} = -w_o\vec{a}_2 \tag{3.4}$$

em que w_o é a velocidade orbital. A velocidade angular de um referencial do satélite, chamado de B, com vetores base $(\vec{b}_1, \vec{b}_2, \vec{b}_3)$, é dada por:

$$\vec{w}_{B/N} = \vec{w}_{B/A} + \vec{w}_{A/N} = \vec{w}_{B/A} - w_o\vec{a}_2 \tag{3.5}$$

em que $\vec{w}_{B/A}$ é a velocidade angular de B em relação a A.

A equação 3.5 permite relacionar os sistemas N, A e B. Porém, para simular a atitude em ângulos de Euler é necessário definir as matrizes de rotação entre os sistemas de referência.

Para descrever a orientação de B com respeito a A em termos dos três ângulos de Euler (ψ, θ, ϕ), considera-se a seqüência de rotações 3-2-1 $(C_1(\psi) \leftarrow C_2(\theta) \leftarrow C_3(\phi))$ de A para B. Note que as setas indicam a ordem das rotações e a disposição das matrizes C_i, $i = 1, 2, 3$, corresponde à posição das mesmas para a multiplicação de matrizes que resulta na matriz de rotação $R_{B/A}$. Para essa seqüência tem-se (WIE, 1998):

$$\vec{b} = R_{B/A}\vec{a} \tag{3.6}$$

$$\begin{aligned} R_{B/A} &= C_1(\psi)C_2(\theta)C_3(\phi) \\ &= \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \\ &= \begin{bmatrix} c\theta c\phi & c\theta s\phi & -s\theta \\ s\psi s\theta c\phi - c\psi s\phi & s\psi s\theta s\phi + c\psi c\phi & s\psi c\theta \\ c\psi s\theta c\phi + s\psi s\phi & c\psi s\theta s\phi - s\psi c\phi & c\psi c\theta \end{bmatrix} \end{aligned} \tag{3.7}$$

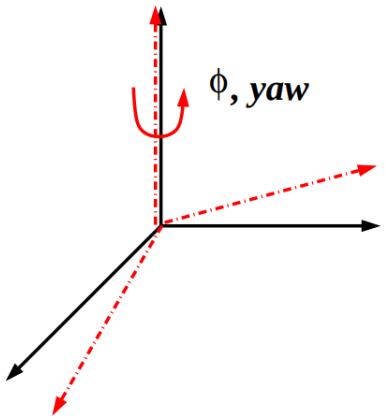
em que $c\beta \equiv cos\beta$ e $s\beta \equiv sen\beta$. As equações 3.6 e 3.7 resultam em:

$$\begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} = \begin{bmatrix} c\theta c\phi & c\theta s\phi & -s\theta \\ s\psi s\theta c\phi - c\psi s\phi & s\psi s\theta s\phi + c\psi c\phi & s\psi c\theta \\ c\psi s\theta c\phi + s\psi s\phi & c\psi s\theta s\phi - s\psi c\phi & c\psi c\theta \end{bmatrix} \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} \tag{3.8}$$

Fisicamente, a seqüência de rotações 3-2-1, com origem no referencial X, Y, Z e destino no referencial X_3, Y_3, Z_3 pode ser descrita como:

a) Rotação em torno do eixo Z de um ângulo ϕ que leva ao sistema X_1, Y_1, Z_1 (Figura 3.11).

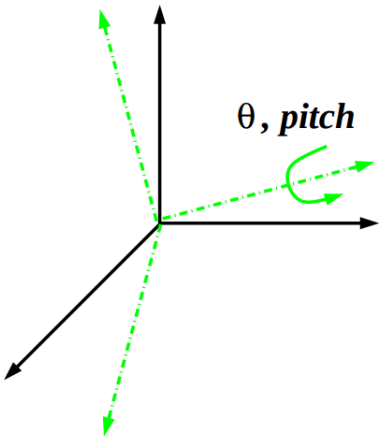
Figura 3.11 - Yaw (ϕ).



Fonte: Adaptado de Ardakani e Bridges (2021).

b) Rotação em torno do eixo Y_1 de um ângulo θ que leva ao sistema X_2, Y_2, Z_2 (Figura 3.12).

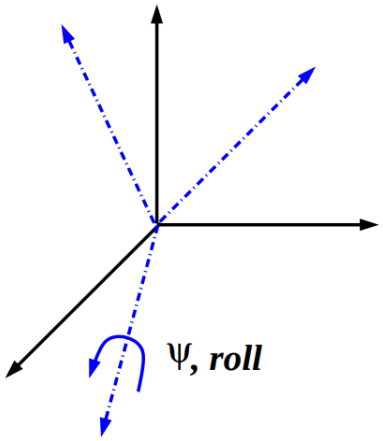
Figura 3.12 - Pitch (θ).



Fonte: Adaptado de Ardakani e Bridges (2021).

c) Rotação em torno do eixo X_2 de um ângulo ψ que leva ao sistema X_3, Y_3, Z_3 (Figura 3.13).

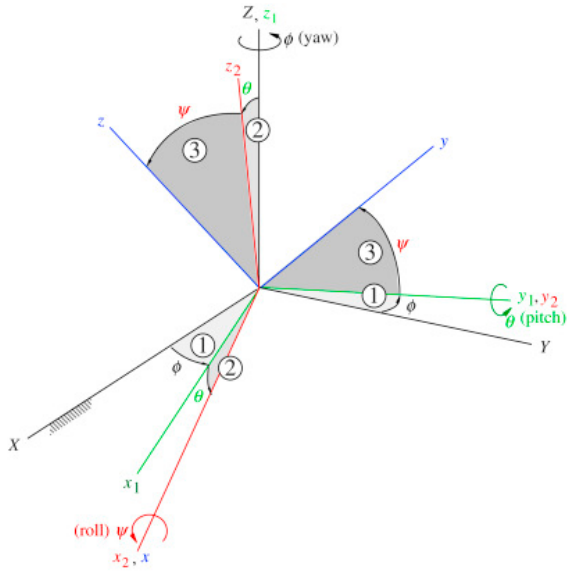
Figura 3.13 - Roll (ψ).



Fonte: Adaptado de Ardakani e Bridges (2021).

As rotações e suas projeções podem ser vistas na Figura 3.14:

Figura 3.14 - Rotação yaw, pitch e roll e suas projeções.



Fonte: Curtis (2014).

O sistema de equações da cinemática, descritos por ângulos de Euler é dependente da seqüência de rotações escolhida para mudar de um sistema de referência para outro. Para o caso em que a diferença angular entre os eixos do sistema no corpo (B) e o sistema orbital (A) é muito pequena, é possível linearizar a matriz de rotação da cinemática. Essa linearização é válida para desacoplar as Leis de Controle e para valores cujas integrais das velocidades angulares sejam menores que 5°. O desacoplamento permite o projeto de uma lei de controle por eixo do sistema no corpo (B). Para as diferenças angulares maiores que 5°, os sinais, mais que os módulos dos ângulos , são mais relevantes para as Leis de Controle.

Para um pequeno ângulo de rotação (α) a seguinte aproximação pode ser considerada:

$$sen(\alpha) \approx \alpha$$
$$cos(\alpha) \approx 1$$

Logo, para a sequência de rotações $C_1(\psi) \leftarrow C_2(\theta) \leftarrow C_3(\phi)$, a velocidade angular de B relativo a A é representada por:

$$\begin{bmatrix} \vec{\omega}_x \\ \vec{\omega}_y \\ \vec{\omega}_z \end{bmatrix}_{B/A} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\psi & sen\psi cos\theta \\ 0 & -sen\psi & cos\psi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} \tag{3.9}$$

Pela equação 3.5 a velocidade angular com que B gira em relação a N, fica então:

$$\begin{bmatrix} \vec{\omega}_x \\ \vec{\omega}_y \\ \vec{\omega}_z \end{bmatrix}_{B/N} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\psi & sen\psi cos\theta \\ 0 & -sen\psi & cos\psi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} - \omega_0 \begin{bmatrix} cos\theta sen\phi \\ sen\psi sen\theta sen\phi + cos\psi cos\phi \\ cos\psi sen\theta sen\phi - sen\psi cos\phi \end{bmatrix} \tag{3.10}$$

O vetor que o termo $(-\omega_0)$ multiplica pode ser obtido da segunda coluna da matriz de rotação da equação 3.8, pois $(-\omega_0)$ só afeta $\vec{a}_2$.

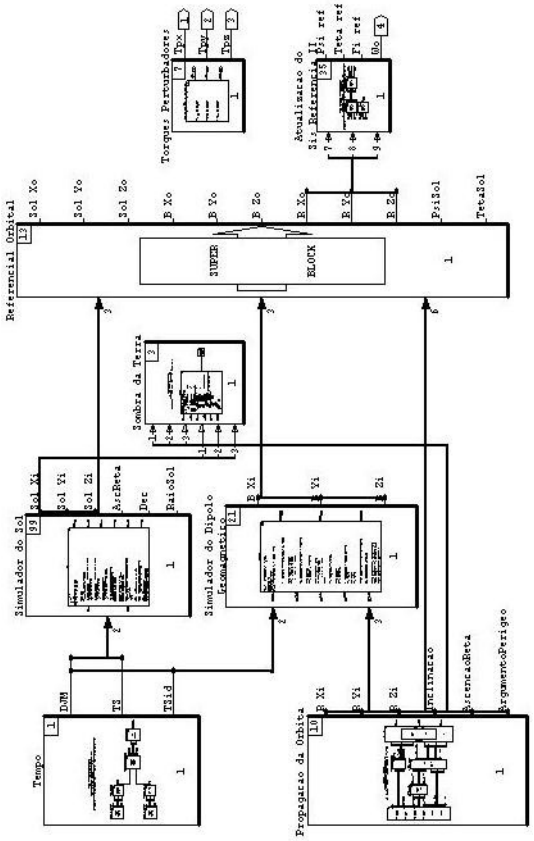
A equação diferencial da cinemática de um corpo rígido em órbita, que representa a seqüência de rotações 3-2-1 é encontrada ao se isolar as derivadas com repeito ao tempo dos ângulos de atitude. Esta fica na forma:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \frac{1}{cos\theta} \begin{bmatrix} cos\theta & sen\psi sen\theta & cos\psi sen\theta \\ 0 & cos\psi cos\theta & -sen\psi cos\theta \\ 0 & sen\psi & cos\psi \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \frac{\omega_0}{cos\theta} \begin{bmatrix} sen^2\theta sen\phi \\ cos\theta cos\phi \\ sen\theta sen\phi \end{bmatrix} \tag{3.11}$$

3.6 Modelo do ambiente espacial

O bloco Ambiente Espacial contém as funcionalidades responsáveis pela propagação da órbita do satélite, cálculo das perturbações e cálculo da velocidade orbital. Esta é utilizada para definir com que velocidade B gira em relação a N e para o cálculo da parte da cinemática do satélite. A Figura 3.15 mostra a modelagem, em diagrama de blocos, das funcionalidades do bloco Ambiente Espacial.

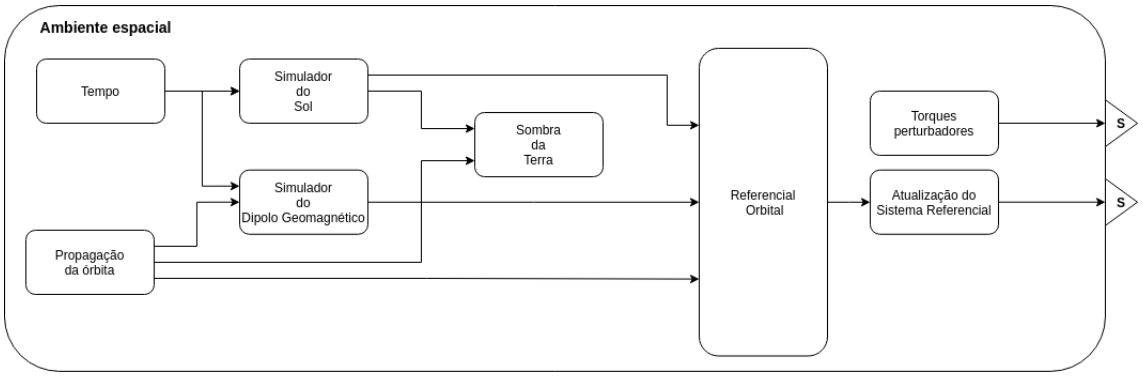
Figura 3.15 - Diagrama de blocos do ambiente espacial do MATRIXx/SystemBuild.



Fonte: Amorim III (2009).

O modelo do Ambiente Espacial pode ser melhor compreendido pelo diagrama funcional da Figura 3.16:

Figura 3.16 - Diagrama de blocos funcional do ambiente espacial. Em que S significa saída.



Fonte: O autor.

3.7 Modelo do sistema de controle de atitude

Segundo Amorim III (2009) o modelo similar ao da PMM de 2001, em condições normais de funcionamento, apresenta um conjunto de sensores e atuadores responsáveis pela manutenção da atitude da plataforma e de sua carga útil.

Os equipamentos ativos em condições normais são apresentados sucintamente a seguir.

3.7.1 Sensores

Conforme Amorim III (2009) em condições normais de funcionamento, o modelo similar ao da PMM de 2001 utiliza três tipos de sensores: o receptor de GPS, a unidade inercial (giroscópios) e o sensor de estrelas. O receptor GPS provê o tempo, a posição e velocidade lineares do satélite, enquanto a unidade inercial é responsável pela obtenção da velocidade angular. O sensor de GPS não foi considerado no modelo herdado. O sensor de estrelas é responsável por fornecer a informação de atitude e, como é o mais preciso dos sensores, é encarregado também por calibrar a unidade inercial.

3.7.1.1 Giroscópios

Segundo Amorim III (2009) Giroscópios formam a base de sensores inerciais para propagação de atitude e controle. Um conjunto de três giros ortogonais entre si é capaz de medir os componentes (ω_x, ω_y, ω_z) da velocidade angular do veículo espacial. Um quarto giroscópio, a um ângulo escolhido em relação aos outros, pode ser utilizado para se evitar um ponto de falha simples.

Gobato (2006) e Moreira (2006) modelaram os giroscópios como um ganho unitário em relação à entrada e saída, acrescido de um ruído aleatório. Este modelo visa imitar aproximadamente o comportamento real do sensor sem elevada complexidade.

As especificações consideradas para sensores inerciais no trabalho de Amorim III (2009) são:

- Alcance dinâmico $= 1^o/s$;
- "Random walk" angular $= 0,01^o/h^{1/2}$ (3σ);
- Deriva $= 0,05^o/h$ (3σ);
- Alinhamento absoluto $= 0,01^o$ (3σ);
- Alinhamento relativo (entre eixos) $= 0,005^o$ (3σ)
- Possibilidade de teste com o equipamento montado;
- Taxa de amostragem $= 10\ Hz$.

3.7.1.2 Sensor de estrelas

Sensores de estrelas determinam a localização e a atitude de um satélite com relação às estrelas catalogadas no entorno do corpo do satélite (ESA, 2020).

Segundo Amorim III (2009) são os mais precisos sensores de referência de uso comum para medição de atitude. É possível obter uma precisão de 1 arco de segundo ou superior.

Gobato (2006) e Moreira (2006) modelaram os sensores de estrelas como um ganho unitário em relação à entrada e saída, acrescido de um ruído aleatório.

As especificações consideradas para sensores de estrelas no trabalho de Amorim III (2009) são:

- Saída na forma de quaternions;
- Precisão no plano do sensor: $\leq 0,0025^o$ (3σ);
- Precisão no eixo normal: $\leq 0,02^o$ (3σ);
- Campo de visão: $\leq 25^o \times 25^o$;
- Número de estrelas seguidas simultaneamente: ≥ 5; com probabilidade $\geq 99\%$ das 5 estarem no campo de visão;
- Para cada estrela seguida: posição (x_i, yi), magnitude (m), número de catálogo (N);
- Taxa de amostragem de pelo menos 1 Hz.

3.7.2 Controlador de atitude

O código do controlador de atitude, que contém as leis de controle, está embarcado no equipamento computador de bordo. O mesmo interage com os sensores e atuadores por meio de sinais de tensão.

Conforme Gobato (2006) e Moreira (2006) devido aos tipos de sensores para condições normais de operação do modelo similar ao da PMM de 2001 (giroscópio e sensor de estrelas), é necessário incluir o cálculo de atitude no código do controlador. O valor de atitude só é disponibilizado diretamente através do sensor de estrelas, porém este é utilizado somente para calibrar os demais equipamentos do satélite; logo, a atitude deve ser obtida indiretamente através da propagação das medidas da velocidade angular oriundas dos giroscópios.

Segundo Amorim III (2009) o controlador utilizado foi considerado discreto, pois os efeitos de quantização do sinal e de atrasos inerentes a um controlador digital não foram analisados. A discretização obedece à regra de Tustin, descrita pela equação 3.12 (FRANKLIN; POWELL, 1981).

$$s \approx \frac{2}{T}\frac{z-1}{z+1} \tag{3.12}$$

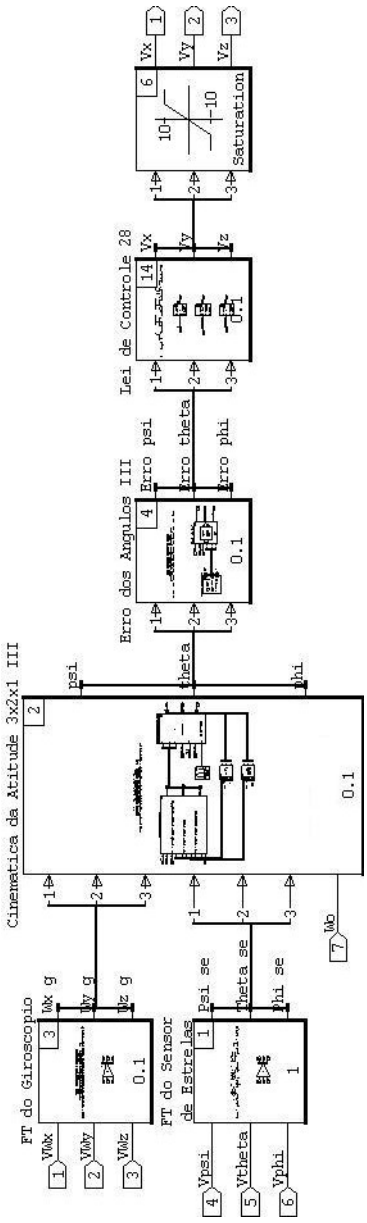
Segundo Amorim III (2009) a taxa de amostragem do controlador foi escolhida para ser a mesma do giroscópio, no caso 10 Hz. O controlador possui uma lei de controle Proporcional, Integral e Derivativa (PID) em relação ao erro da atitude para cada um dos ângulos de atitude do satélite.

Gobato (2006) e Moreira (2006) calcularam os ganhos do controlador. Para isso definiram o ganho integral (K_i) como 1 para cada eixo do satélite; e, baseados nas especificações de amortecimento (ς) de 0.7 e tempo de subida (T_r) de 100 segundos calcularam o restante. Então, os ganhos para os eixos X, Y, Z são:

$$\begin{aligned} K_i &= [1 \quad 1 \quad 1] \\ K_p &= [40{,}593 \quad 51{,}785 \quad 44{,}354] \\ K_d &= [454{,}11 \quad 556{,}93 \quad 488{,}66] \end{aligned}$$

O modelo do Controlador de Atitude baseado em ângulos de Euler se encontra na Figura 3.17

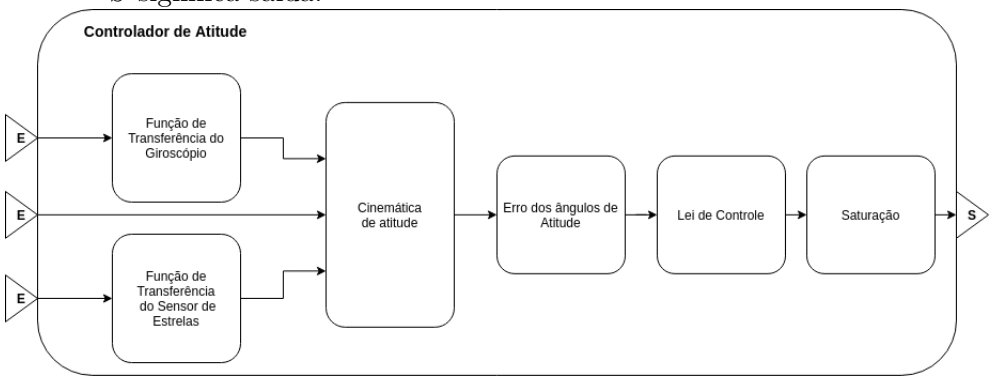
Figura 3.17 - Diagrama de blocos do controlador baseado em ângulos de Euler feito no MATRIXx/SystemBuild.



Fonte: Amorim III (2009).

O modelo do Controlador de Atitude pode ser melhor compreendido pelo diagrama funcional da Figura 3.18:

Figura 3.18 - Diagrama de blocos funcional do controlador. Em que E significa entrada e S significa saída.



Fonte: O autor.

3.7.3 Atuadores

Segundo Amorim III (2009) os atuadores utilizados em condições normais de funcionamento são um conjunto de três rodas de reação dispostas cada uma sobre os três eixos principais de inércia do satélite e uma quarta inclinada em relação aos demais eixos. A quarta roda produz componentes de torque sobre os três eixos. Isto permite a atuação em qualqer dos eixos no caso da ocorrência de falha de qualquer das outras rodas. O trabalho herdado não considera a presença da quarta roda de reação.

Gobato (2006) e Moreira (2006) desenvolveram o modelo de rodas de reação baseados na função de transferência presente no trabalho de mestrado de Souza (1981). A função de transferência possui a seguinte forma:

$$\frac{T}{V} = \frac{K_v T_v s}{(1 + T_v s)} \tag{3.13}$$

$$\frac{\omega}{T} = \frac{1}{J_w s} \tag{3.14}$$

em que T é o torque produzido pela roda, V é a tensão de entrada do motor da roda, K_v é o ganho, T_v é a constante de tempo, ω é a velocidade angular e J_w é o momento de inércia da roda de reação.

Para o cálculo das constantes K_v e T_v devem ser utilizadas as seguintes equações:

$$\begin{aligned} T_v &= \frac{J_w \omega_{max}}{T_{max}} \\ K_v &= \frac{T_{max}}{V_{max}} \end{aligned} \tag{3.15}$$

em que ω_{max} é a velocidade angular máxima da roda de reação, T_{max} é o torque máximo no motor e V_{max} é a tensão máxima na entrada do motor da roda de reação.

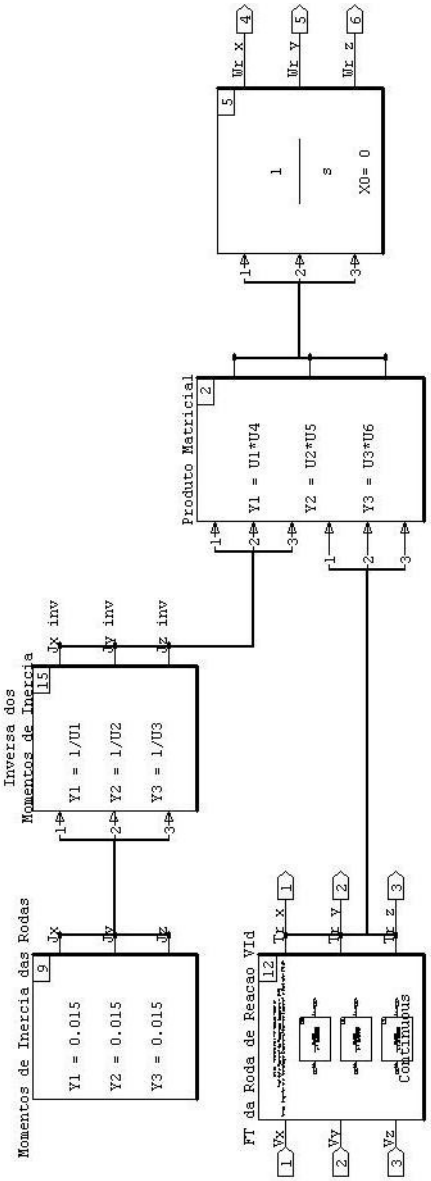
Gobato (2006) e Moreira (2006) consideraram rodas de reação hipotéticas em seu trabalho, com constantes de tempo de aproximadamente 20 s, e o torque máximo de aproximadamente $0,6$ Nm. O trabalho de Amorim III (2009) foi focado no desenvolvimento de simuladores, portanto um modelo atualizado de rodas de reação não foi explorado.

Os cálculos realizados por Gobato (2006) e Moreira (2006) resultaram nos parâmetros: $T_v = 20\ s$ e $K_v = 0,06$. A função de transferência da roda de reação fica então:

$$\frac{T}{V} = \frac{1,2s}{(1+20s)} = \frac{0,06s}{(s+0,05)} \tag{3.16}$$

O modelo dos atuadores (rodas de reação) pode ser visto na Figura 3.19:

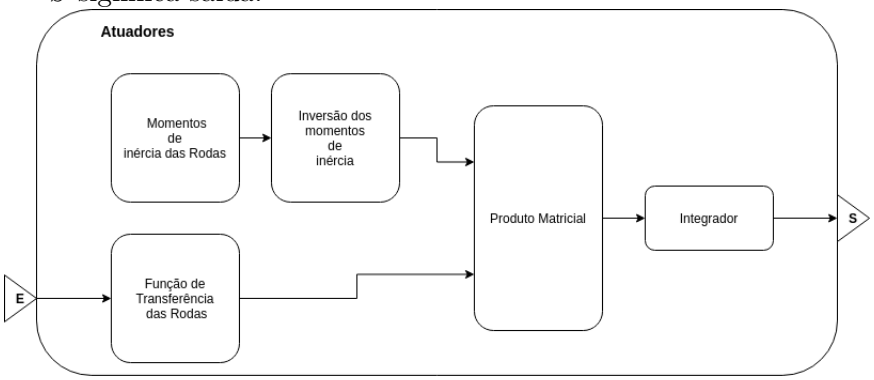
Figura 3.19 - Diagrama de blocos dos atuadores (Rodas de reação) feito no MatrixX/SystemBuild.



Fonte: Amorim III (2009).

O modelo dos atuadores (rodas de reação) pode ser melhor compreendido pelo diagrama funcional da Figura 3.20:

Figura 3.20 - Diagrama de blocos funcional dos atuadores. Em que E significa entrada e S significa saída.

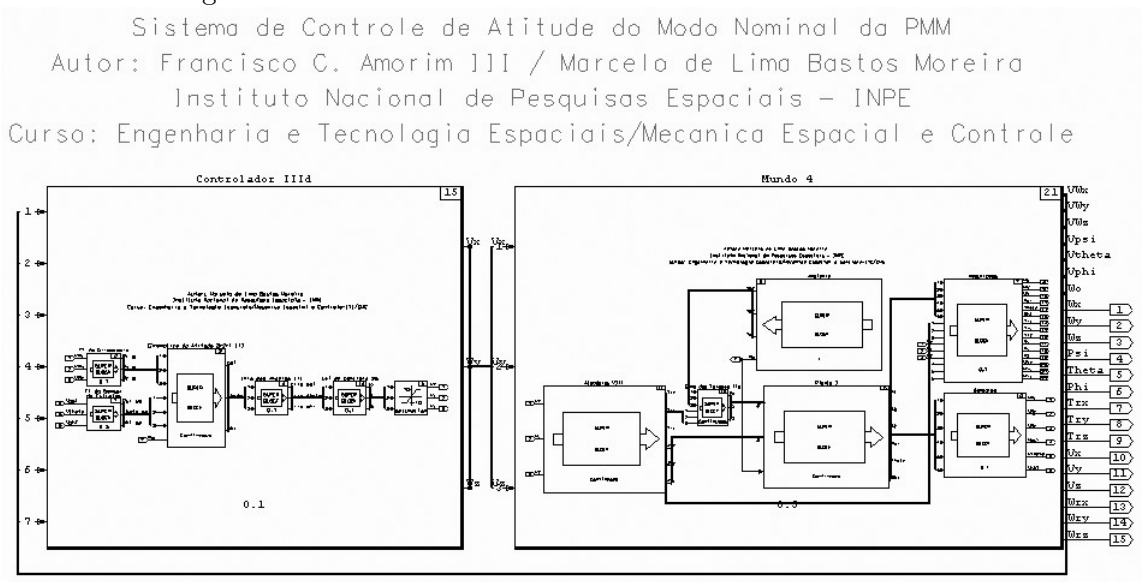


Fonte: O autor.

3.8 Sistema completo

O sistema completo composto por controlador de atitude, atuadores, sensores, ambiente espacial, cinemática e dinâmica do satélite utilizado por Amorim III (2009) pode ser visto na Figura 3.21:

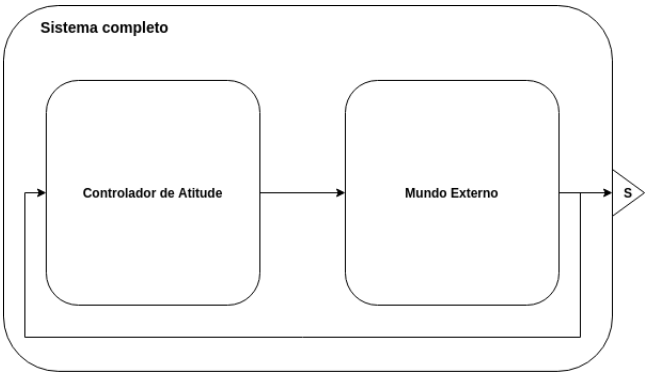
Figura 3.21 - Diagrama de blocos, representado no MatrixX/SystemBuild, do modelo reorganizado dividido em: Controlador de Atitude e Mundo.



Fonte: Amorim III (2009).

O modelo do sistema completo pode ser melhor compreendido pelo diagrama funcional da Figura 3.22:

Figura 3.22 - Diagrama de blocos funcional do sistema completo. Em que S significa saída.



Fonte: O autor.

3.8.1 Requisitos de tempo

Segundo Amorim III (2009) os seguintes requisitos de tempo foram assumidos para o sistema completo:

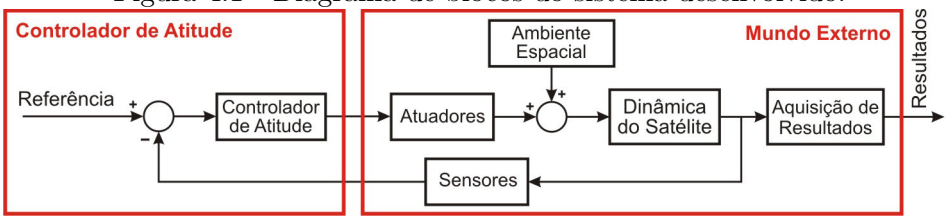
- Período de amostragem do sistema = 100 ms
- Tempo de aquisição de novas medidas (sensores) $\leq$ 10 ms
- Tempo para cálculo da tensão de controle (controlador) $\leq$ 10 ms
- Tempo para fornecimento do torque de controle (atuadores) $\leq$ 40 ms
- Tempo para atualização da atitude (planta) $\leq$ 30 ms

Segundo Amorim III (2009) o período de amostragem do sistema foi escolhido para ser igual ao período de amostragem do giroscópio. Este período de amostragem é mais do que suficiente pois a PMM de 2001 é uma planta com constante de tempo elevada (dinâmica lenta). Além do mais, a especificação de reposicionamento exige que para uma mudança de 30 graus o tempo necessário seja de no máximo 3 minutos (180 segundos). Logo, a especificação de período de amostragem atende com uma margem de múltiplas ordens de magnitude o intervalo de tempo exigido pela manobra.

4 SISTEMA DE SIMULAÇÃO E MIGRAÇÃO

Amorim III (2009) desenvolveu em seu trabalho, baseado nas dissertações de Gobato (2006) e Moreira (2006), um simulador de Software de Controle de Atitude para um modelo similar ao da PMM de 2001, constituído de duas partes principais. A primeira é o Controlador de Atitude discreto, responsável pelo controle de atitude do satélite e a segunda, o Mundo Externo, responsável por simular tudo aquilo que é externo ao controlador (sensores, atuadores, ambiente espacial, cinemática e dinâmica da planta). A topologia em diagrama de blocos pode ser vista na Figura 4.1:

Figura 4.1 - Diagrama de blocos do sistema desenvolvido.



Fonte: Amorim III (2009).

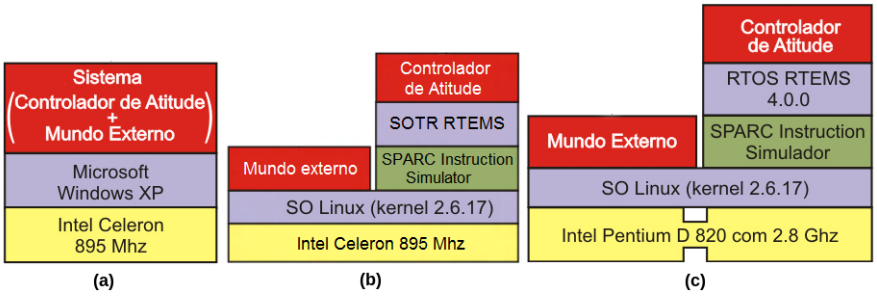
O presente trabalho herda a topologia e a divisão das partes do sistema e difere apenas na implementação da simulação.

4.1 Implementações anteriores Amorim III (2009)

Amorim III (2009) em seu trabalho realizou cinco implementações diferentes de simulador. A primeira, realizada em um ambiente MatrixX/SystemBuild instalado sobre um SO Windows XP e um PC Intel Celeron 895 GHz, a fim de apenas escolher e testar as leis de controle. A segunda, realizada em um ambiente MatrixX/AutoCode instalado sobre um SO Windows XP e um PC Intel Celeron 895 GHz., a fim de apenas gerar e testar o código em C ANSI. A terceira, foi realizada em uma plataforma computacional iHawk (Xeon temporizado) com SOTR RedHawk Linux (RedHat temporizado). A quarta e a quinta foram realizadas em um SO Linux 2.6.17 com o pacote de ferramentas ERC32CSS (ERC32 GNU Cross-Compiler System). Este pacote fornece um ambiente de configuração para simular o processador ERC32 (SIS) suportando o SOTR RTEMS 4.0.0. A quarta e a quinta implementações diferem apenas no processador: uma foi realizada em um processador Intel Celeron 895 MHz de um núcleo; e a outra foi realizada em um processador Intel Pentium

D 820 2.8GHz de dois núcleos, respectivamente. Para este trabalho são relevantes a primeira e a segunda implementações em ambiente MatrixX/Windows e a quarta e quinta implementações que utilizaram o pacote ERC32CSS. As implementações podem ser vistas na Figura 4.2:

Figura 4.2 - Implementações realizadas por Amorim III (2009) relevantes para este trabalho. (a) Primeira/segunda implementações, (b) quarta implementação e (c) quinta implementação.



Fonte: Amorim III (2009).

Na primeira implementação de Amorim III (2009) (Figura 4.2a) o sistema de simulação completo (Mundo Externo e Controlador de Atitude) foi executado em malha fechada no próprio ambiente MatrixX/SystemBuild a fim de se testar leis de controle e obter uma resposta de referência para as implementações subsequentes. Para as demais implementações Amorim III (2009) codificou as duas partes principais em dois programas em linguagem C separados. Um código responsável pelo Controlador de Atitude e outro código responsável pelo Mundo Externo. Na segunda implementação, os códigos gerados para o Controlador de Atitude e Mundo Externo são compilados e executados em malha fechada no próprio Windows XP a fim de se analisar a resposta do simulador gerado sem separação de ambiente computacional. A segunda implementação possui a mesma estrutura da primeira implementação apresentada na Figura 4.2. As respostas da primeira e segunda implementações são relevantes pois servem como referências para as demais. A quarta e a quinta implementações são relevantes, pois estas incluem a simulação do processador ERC32 em seus níveis de abstração. A diferença entre os processadores não é significativa no caso, pois a arquitetura funcional da implementação é mais significativa para o caso deste trabalho. O funcionamento destas duas implementações pode ser resumido da seguinte forma: O programa de simulação do Mundo Externo é executado sobre o

SO Linux, enquanto o Controlador de Atitude é executado sobre o SOTR RTEMS. Este é executado sobre um Simulador de instruções SPARC (SIS). O SIS, por sua vez, é executado sobre o SO Linux, simulando a arquitetura do processador ERC32 (SPARC). Desta forma é possível verificar se o código do controlador é executado como esperado na arquitetura SPARC. Esta abordagem permite realizar um teste de necessidade do código. Isto significa que, se o código não executar corretamente na arquitetura SPARC, ele não será executado no ERC32; porém, se o mesmo for executado corretamente, isto não é suficiente para garantir que, ele será executado corretamente no processador. A vantagem desta abordagem, é a capacidade de verificação incremental do projeto; em outras palavras, é possível realizar testes no código gerado antes de implementá-lo em um processador real. Consequentemente, isto reduz o custo de projeto e permite a produção de código melhor.

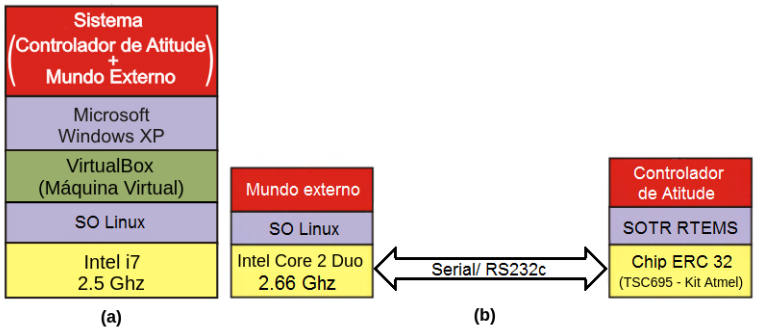
4.2 Implementação atual

O trabalho atual realizou três implementações diferentes de simulador. A primeira foi realizada em um ambiente MatrixX/SystemBuild instalado sobre um SO Windows XP instalado sobre uma máquina virtual VirtualBox por sua vez instalada em PC com SO Linux Mint com processador Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz e 16 GB de RAM. Esta foi realizada de forma a replicar a primeira implementação de Amorim III (2009), com a finalidade de gerar dados de simulação para servir referência para as outras implementações. A segunda foi realizada em um ambiente MatrixX/AutoCode instalado na mesma máquina virtual e computador da primeira implementação de forma a replicar a segunda implementação de Amorim III (2009). Nesta a finalidade é gerar, testar e comparar o código gerado com o de Amorim III (2009). Assim como no caso de Amorim III (2009), a primeira e segunda implementações possuem a mesma estrutura.

A terceira consistiu em implementar o programa de simulação do Controlador de Atitude em um processador ERC32 e preservar o programa de simulação do Mundo Externo em um computador com sistema operacional Linux Mint. A técnica de projeto PIL é utilizada; desta forma, a execução do programa está sujeita não somente à arquitetura lógica do processador, porém também à arquitetura física do mesmo. A execução do código fica subordinada às particularidades do processador citadas na Subseção 2.2.3. A migração do programa do Controlador de Atitude implica a necessidade de comunicar o processador ERC32, que se encontra no kit da Atmel, com o computador PC em que o programa de simulação do Mundo Externo será executado. A comunicação é realizada através de uma porta Serial UART com protocolo

RS-232 como mostra a Figura 4.3;

Figura 4.3 - Implementações realizadas neste trabalho. (a) primeira/segunda implementações, e (b) terceira implementação.



Fonte: O autor.

No trabalho atual, o autor adaptou os códigos do simulador do Controlador de Atitude e do Mundo Externo para que a terceira implementação da Figura 4.3 fosse realizada. A Adaptação foi realizada de forma a apenas introduzir uma interface no código para que a comunicação entre as partes fosse possível. Portanto não houve alteração na funcionalidade principal do código, de forma que não comprometesse a comparação com os resultados de Amorim III (2009).

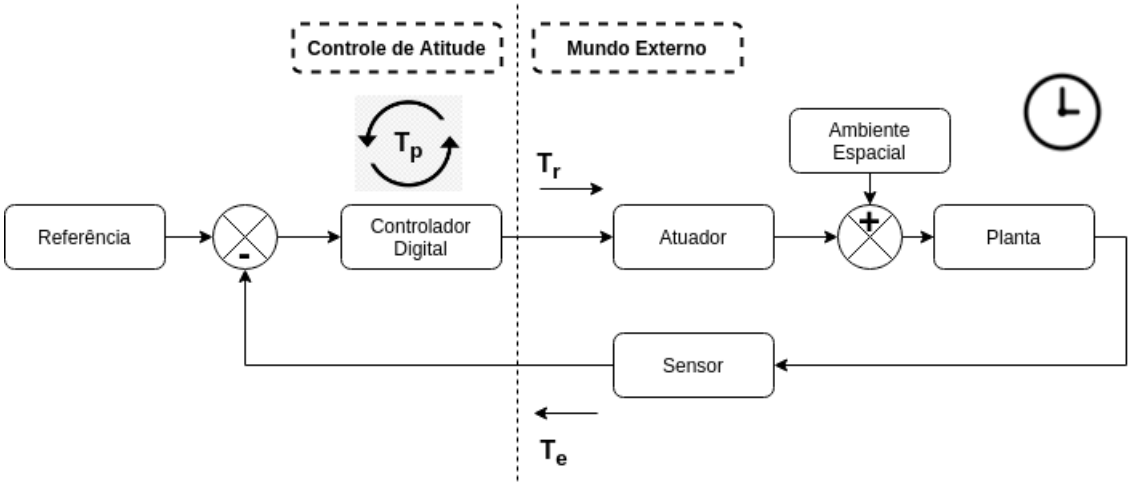
4.2.1 Experimento temporal

Sistemas de tempo real necessitam que os tempos de execução das tarefas consideradas sejam conhecidos (ou, ao menos, sua soma). É comum se considerar sua função densidade de probabilidade ou o pior caso de tempo de execução possível para uma tarefa. Portanto, um experimento ("ping-pong") com 10.000 medidas foi realizado para saber o tempo de execução das tarefas de envio ("ping"), processamento e recebimento ("pong") de dados entre as partes. O experimento consistiu em medir o intervalo de tempo que ($\delta t = t_2 - t_1$) desde o início (t_1) do envio dos dados dos sensores para o Controlador de Atitude (RTEMS/ERC32), serem processados por este, até o fim (t_2) do recebimento dos dados pelo Mundo Externo (Linux/PC). A medida de tempo foi realizada pela rotina *clock_gettime* da linguagem C ANSI no software do Mundo Externo, isto é, onde se encontra o modelo da planta. Desta forma a medida de tempo reflete como a planta "enxerga/percebe"temporalmente o controlador. O tempo de processamento do Controlador de Atitude foi medido

indiretamente, pois a base de tempo deste é diferente da base de tempo do Mundo Externo. Como a operação de processamento dos dados ocorre após o envio dos dados do Mundo Externo e antes do recebimento dos dados pelo Mundo Externo, a medição de tempo entre o envio e o recebimento agrega o tempo de processamento.

O experimento pode ser melhor compreendido na Figura 4.4:

Figura 4.4 - Modelo de percepção temporal da planta.



Fonte: O autor.

Em que: T_r é tempo de recebimento dos dados, T_e é o tempo de envio dos dados e T_p é o tempo de processamento dos dados.

O período de amostragem tem um valor mínimo dado por:

$$T_{amostragem} \geqslant T_{amostragem\_min.} = T_r + T_p + T_e = \delta t = t_2 - t_1$$

Em uma amostra de 10.000 medidas, os valores obtidos para a média, desvio padrão, e maior valor do período de amostragem mínimo ($T_{amostragem\_min.}$) foram:

média	desvio	maior
54.4522 ms	0.1066 ms	54.5589 ms

Seguindo Amorim III (2009), neste trabalho, adotou-se $T_{amostragem} = 100$ ms, para

acomodar o tempo de processamento do Mundo Externo.

4.2.2 Funcionamento lógico-informacional

O simulador compreende as duas partes principais: Controlador de Atitude e Mundo Externo. A simulação pode ser dividida em três fases: inicialização, execução e encerramento.

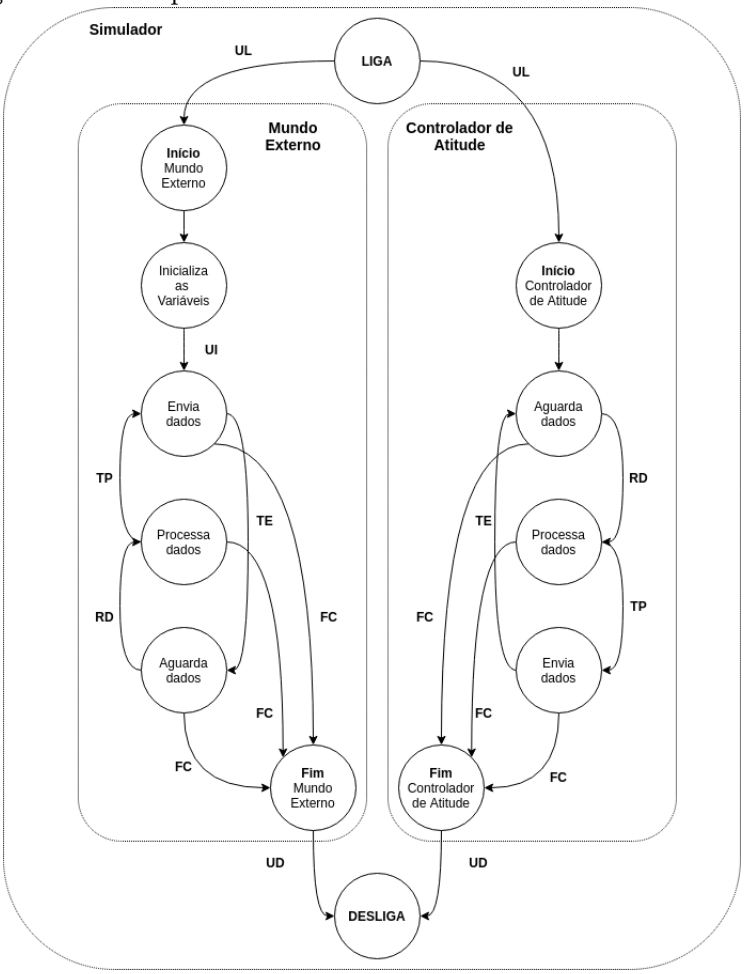
1) **Inicialização**: Nesta fase, o Mundo Externo inicializa às variáveis de estado com um "zero de máquina"(um número muito pequeno reconhecido pela máquina) e o Controlador de Atitude fica em estado de espera dos dados provenientes do mundo externo.

2) **Execução**: Nesta fase, o usuário inicia o ciclo de simulação e o primeiro conjunto de valores da saída da planta prevenientes do Mundo Externo são enviados ao Controlador de Atitude. Este, em seguida, envia um conjunto de valores de sinal de controle de volta ao Mundo Externo. Este ciclo se repete até a fase de encerramento da simulação.

3) **Encerramento**: Nesta fase, o ciclo de simulação atinge o limite de ciclos definido pelo usuário. O Mundo Externo o Controlador de Atitude param de enviar e receber dados, e seguida o programa do simulador é encerrado.

O simulador é melhor representado pela máquina de estados da Figura 4.5:

Figura 4.5 - Máquina de estados do funcionamento do simulador.



Fonte: O autor.

Em que as transições são:

- UL: Usuário liga o sistema.
- UI: Usuário inicia a simulação (Somente no Mundo Externo).
- TP: Ativa quando o processamento de dados termina.
- RD: Ativa quando o recebimento dos dados termina.
- TE: Ativa quando o envio dos dados termina.
- FC: Ativa quando o simulador atinge o fim do ciclo de simulação definido pelo usuário.

- UD: Usuário desliga o sistema.

O ciclo de execução do sistema desenvolvido no presente trabalho, pode ser descrito nos seguintes passos:

0) O Mundo Externo no PC com SO Linux inicializa as variáveis de estado da dinâmica da planta.

1) O Mundo Externo no PC com SO Linux envia os dados via porta serial (RS232) para o processador ERC32 com o Controlador de Atitude.

2) O processador ERC32 com o Controlador de Atitude recebe os dados.

3) O processador ERC32 com o Controlador de Atitude processa os dados recebidos.

4) O processador ERC32 com o Controlador de Atitude envia os dados processados para o software no PC com SO Linux.

5) O Mundo Externo no PC com SO Linux recebe os dados.

6) O Mundo Externo no PC com SO Linux processa os dados recebidos.

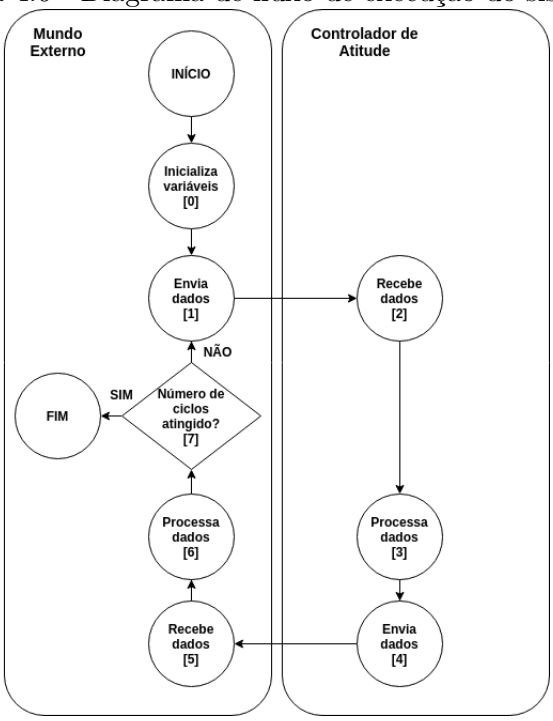
7) O Mundo Externo no PC com SO Linux avalia o número de ciclos de simulação executados. Caso tenha sido atingido a simulação termina.

O ciclo se repete enquanto o número de execuções definido pelo usuário não for atingido.

É importante notar que é no processamento de dados do Mundo Externo que ocorre o passo de integração da dinâmica da planta. No processamento de dados do Controlador de Atitude ocorre a computação das Leis de Controle.

O ciclo de execução da simulação pode ser melhor compreendido pelo diagrama da Figura 4.6:

Figura 4.6 - Diagrama de fluxo de execução do sistema.



Fonte: O autor.

A abordagem temporal de execução utilizada para o SCA é a Executiva Cíclica. Nesta, o agendador executa um conjunto de tarefas não-preemptíveis (não podem ser interrompidas) segundo uma tabela que descreve o momento de execução de cada tarefa. Esta abordagem foi suficiente para este trabalho já que há apenas três tarefas a serem executadas sequencialmente, estas são: envio, processamento e recebimento de dados. O período de amostragem mínimo para esta abordagem é considerado exatamente como no experimento temporal; em outras palavras, é a soma dos tempos de envio, processamento e recebimento dos dados. O requisito (3.8.1) exige um período de amostragem de 100 ms para o sistema, logo o período atingido no experimento atende o requisito.

4.2.3 Detalhes de implementação

Sistemas em tempo real possuem muitos detalhes que afetam diretamente o seu desempenho ou até mesmo sua realizabilidade. Temporizar e sincronizar as partes do sistema enquanto as mesmas se comunicam entre sí é uma tarefa de alta dificuldade. A dificuldade reside no fato de que as tarefas de temporizar, sincronizar e comunicar afetam umas às outras. Um erro de comunicação pode entregar os dados

fora de uma intervalo de tempo aceitável e consequentemente violar a temporização e dessincronizar as partes. Um erro de temporização pode fazer com a que a parte de comunicação não receba ou envie os dados no intervalo correto de tempo e pode dessincronizar as partes. Um erro de sincronia pode fazer com que as bases de tempo das diferentes partes desviem em relação a uma tolerância permitida, e consequentemente causar erro de comunicação e até mesmo mesclar dados de intervalos de tempo diferentes em um mesmo intervalo. Portanto, alguns métodos e técnicas foram desenvolvidos neste trabalho a fim de se mitigar os problemas citados anteriormente. Estes são detalhados nos tópicos adiante.

4.2.3.1 Comunicação UART por interrupção

A comunicação UART foi implementada por meio de funções de escrita e leitura de bytes padrão do sistema (em linguagem C) em conjunto com interrupções do processador ERC32 para realizar transmissão e recebimento dos dados. Estas funções escrevem e leem respectivamente bytes de uma porta, neste caso serial. A rotina de interrupção do ERC32 serve para aguardar a chegada de novos bytes (dados) na porta serial do processador. Esta implementação foi necessária porque não foi encontrado um driver que tivesse funções para envio e recebimento de dados via Serial UART para o RTEMS.

4.2.3.2 Mapeamento de caracteres

As funções de escrita (*write*) e leitura (*read*) do RTEMS realizam um mapeamento não desejado de bytes em caracteres. Isto significa que os bytes enviados ou recebidos não eram percebidos com seus valores absolutos pelo RTEMS, mas eram mapeados em caracteres. Consequentemente, os caracteres poderiam ser interpretados como um comando que resultava na interrupção da comunicação. A sequência de caracteres então foi remapeada com caracteres adicionais, de forma a evitar o envio dos caracteres que causam problema na comunicação.

4.2.3.3 Habilitação e desabilitação da interrupção

O ERC32 possui uma interrupção para informar quando há um novo conjunto de bytes no registrador UART. Esta ativa independentemente do sentido dos bytes, isto é, se são de entrada (de outro ambiente) ou de saída (destinados a outro ambiente). Este comportamento peculiar é indesejado, pois quando dados são enviados a rotina de interrupção é ativada, o que faz com que o processador ative sua própria interrupção. Desta forma a execução do processador ERC32 fica restrita somente ao

contexto de interrupção. Consequentemente, o código principal do Controlador de Atitude nunca é executado. Portanto, um sistema de sincronismo foi desenvolvido de forma que a interrupção se desative assim que receber dados; e se reative após o processamento e envio dos dados pelo código principal do Controlador de Atitude.

4.2.3.4 Sincronização da entrega dos dados

A correta execução de uma simulação depende da causalidade de suas partes; isto é, cada bloco ou subsistema que compõe o modelo e, consequemente seu código, possuem uma ordem determinada de recebimento e envio de dados. Isto significa que a implementação deve garantir que a ordem dos resultados das partes seja preservada e que os mesmos sejam computados durante o mesmo passo de integração. Logo, o protocolo de comunicação deve ser adequadamente configurado de forma que a causalidade da informação entre os ambientes que se comunicam seja garantida. Portanto, uma máquina de estados foi implementada a fim de sincronizar a transmissão e recebimento dos dados entre os ambientes. A mesma pode ser vista na Figura 4.5.

4.2.3.5 Inversão de memória

O ERC32 (Controlador de Atitude) armazena os dados na memória de forma invertida em relação ao computador de simulação (Mundo Externo), isto significa, que um endereço na memória do ERC32 que aponta para o início de um vetor, no computador apontaria para o fim do vetor e vice-versa. Um vetor com tamanho de 10 bytes, por exemplo, tem endereço de byte de 0 a 9, sendo 0 o primeiro byte e 9 o último byte. No computador de simulação o endereço 0 do vetor corresponde ao endereço 9 no processador ERC32, e o endereço 9 do computador de simulação corresponde ao endereço 0 no processador ERC32. Portanto, um sistema de inversão de dados na memória foi desenvolvio a fim de inverter a ordem dos vetores de dados antes de enviá-los ao ERC32 e ao recebê-los do mesmo.

4.2.3.6 Formatação de dados

As variáveis utilizadas nos códigos gerados são de dupla precisão, isto é utilizam 64 bits (8 bytes) para sua representação. Consequentemente, as variáveis possuem muitas casas decimais. Uma formatação indevida dos dados para armazenamento poderia não apresentar várias ordens de casas decimais ou até mesmo zerar valores. Isto acarretaria uma análise incorreta dos resultados da simulação. Neste caso, o analista poderia errôneamente atribuir problemas ao simulador ou à configuração

da simulação quando, na verdade o erro, seria no armazenamento. Portanto, uma formatação que preservasse o máximo de casas decimais possíveis foi definida para evitar erros de interepretação dos dados.

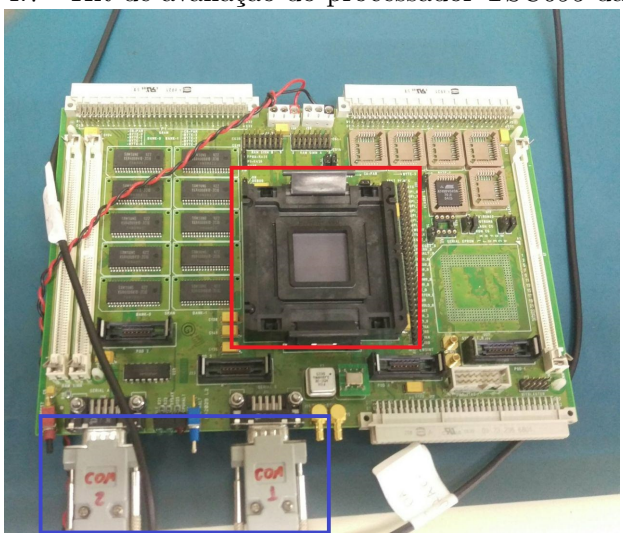
4.2.4 Sistema físico

O sistema físico utilizado para realização deste trabalho é constituído por um Kit de avaliação do TSC695, um PC Intel(R) i7 CPU @ 2.50GHz, 16 GB de RAM, com SO Linux 19 Tara,um PC Intel Core2 Duo @2.66 GHz, 4 GB de RAM, com SO Linux Mint 19.1 Tessa e um PC Intel(R) Core(TM)2Quad @2.40 GHz, 5GB de RAM, com SO Windows 7 Professional.

4.2.4.1 Kit de avaliação TSC695 da Atmel

A Atmel desenvolveu um kit para avaliação do processador TSC695. O kit consiste de uma placa eletrônica que serve de plataforma para alimentação e comunicação com o processador. O kit tem vários barramentos para servir diferentes periféricos (VME, RAM, ROM e FPGA) e duas portas seriais principais, uma para programação e depuração e outra para console/terminal do chip. A Figura 4.7 destaca o processador em vermelho e as portas seriais em azul.

Figura 4.7 - Kit de avaliação do processador TSC695 da Atmel.



Fonte: O autor.

4.2.4.2 PC com SO Linux Mint 19 Tara

Este computador (PC) foi utilizado para realizar a primeira e segunda implementação deste trabalho. Foi utilizado para simulação no ambiente MatrixX/SystemBuild e geração de código no MatrixX/AutoCode.

Características do PC:

- Processador: Intel(R) Core(TM) i7-4710HQ @ 2.50GHz
- Sistema Operacional: Linux Mint 19 Tara
- Memória RAM: 16GB

Características da Máquina virtual:

- Nome: Oracle VM Virtual Box
- Sistema Operacional: Windows XP (32 bits)
- Memória RAM: 4 GB

4.2.4.3 PC com SO Linux Mint 19.1 Tessa

Este computador (PC) foi utilizado para executar o código Mundo Externo do sistema de simulação. O início do ciclo de execução ocorre neste computador.

Características do PC:

- Processador: Intel Core2 Duo @2.66 GHz
- Sistema Operacional: Linux Mint 19.1 Tessa
- Memória RAM: 4GB

4.2.4.4 PC com SO Windows 7 Professional

Este computador (PC) foi utilizado para o desenvolvimento do código para o SO RTEMS a ser executado no processador ERC 32 (TSC695). O código foi desenvolvido no ambiente de programação Eclipse que executa dentro de uma máquina virtual com Linux Mandriva.

Características do PC:

- Processador: Intel(R) Core(TM)2Quad @2.40 GHz
- Sistema Operacional: Microsoft Windows 7 Professional
- Memória RAM: 5GB

Características da Máquina virtual:

- Nome: VMWare
- Sistema Operacional: Linux Mandriva 2010.2
- Memória RAM: 1.5 GB

4.2.4.5 Sistema de bancada

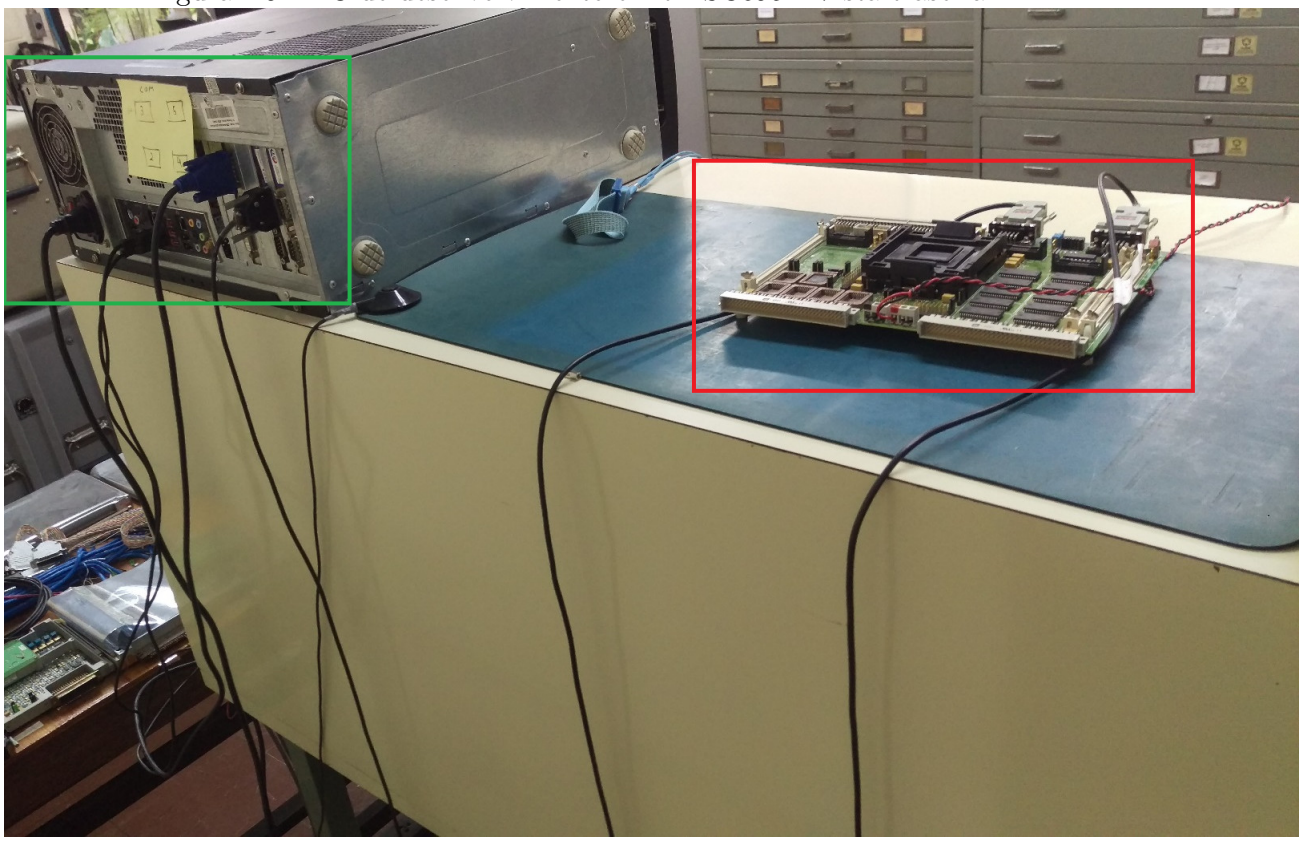
O sistema físico de bancada consiste do Kit de avaliação TSC695 da Atmel, o PC com SO Linux Mint 19.1 Tessa e o PC com SO Windows 7 Professional. O PC com SO Linux Mint 19 Tara não faz parte do sistema de bancada pois não foi utilizado para o desenvolvimento do código para o processador TSC695 (ERC32) nem para simulação em malha fechada com o processador. Além do mais, este PC não faz parte do laboratório onde está a bancada.

O Sistema físico de bancada pode ser visto nas Figuras 4.8, 4.9 e 4.10 a seguir em diferentes vistas. É possível também, notar os cabos e conectores para comunicação serial. Os componentes estão destacados por cores. O kit TSC695 com processador ERC32 está destacado por um retângulo vermelho, O PC com SO Linux Mint 19.1 Tessa para execução de simulação por um retângulo azul e o PC com SO Windows 7 Professional para desenvolvimento destacado por um retângulo verde.

Figura 4.8 - Sistema Físico Completo - Vista Frontal.

Fonte: O autor.

Figura 4.9 - PC de desenvolvimento e kit TSC695 - Vista traseira.



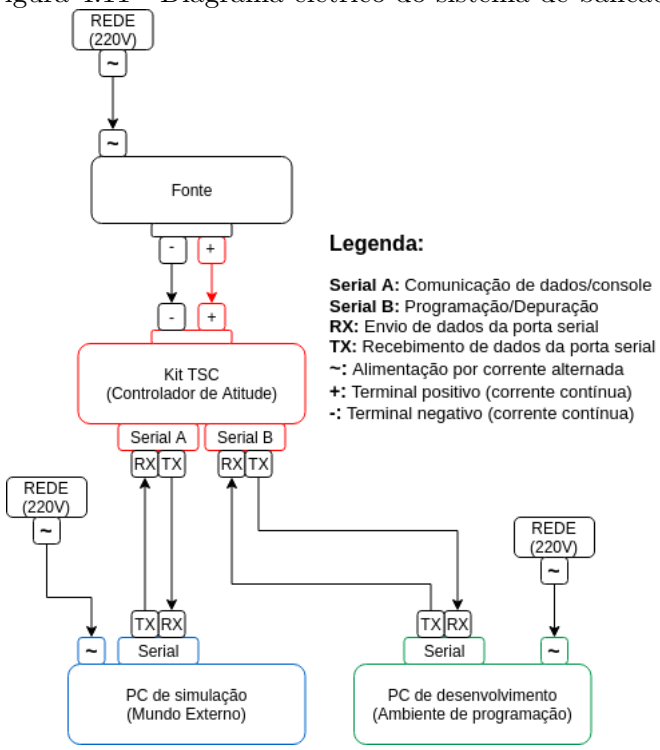
Fonte: O autor.

Figura 4.10 - PC de execução - Vista traseira.

Fonte: O autor.

Um diagrama elétrico do sistema de bancada pode ser visto na Figura 4.11:

Figura 4.11 - Diagrama elétrico do sistema de bancada.



Fonte: O autor.

5 ANÁLISE E RESULTADOS

As simulações realizadas por Amorim III (2009) foram puramente computacionais. Este trabalho se propôs a inserir um processador (ERC32) real na malha de controle. Os resultados obtidos são baseados nas mesmas condições propostas por Amorim III (2009).

5.1 Manobra de referência

A Manobra de atitude utilizada como referência para a simulação é a mesma utilizada no trabalho de Amorim III (2009). A manobra consiste em levar o ângulo de atitude ψ (Psi), de 30º para 0º $\pm$ 0.05º e permanecer em 0º/s $\pm$ 0.001º/s. Toda a manobra deve ocorrer em menos de 180 segundos.

5.2 Gráficos

As figuras 5.1-5.24 a seguir apresentam gráficos das comparações de variáveis de interesse do sistema. No total são 15 variáveis de interesse, porém nesta seção somente os gráficos de 4 variáveis serão apresentadas, pois estas apresentam as características mais relevantes para o sistema. Os gráficos destas e das demais podem ser vistos no Apêndice A.

Os gráficos são apresentados baseado nas seguintes implementações:

- Gerados no ambiente MatrixX/SystemBuild executado sobre um SO Windows XP e um PC Intel Celeron 895 MHz (implementação 1 anterior, Figura 4.2a) e comparados aos gerados no simulador gerado pelo ambiente MatrixX/AutoCode, compilado e executado em um SO Windows XP em um PC Intel Celeron 895 MHz (implementação 1 anterior, Figura 4.2a), ambos feitos por Amorim III (2009).

- Gerados no ambiente MatrixX/SystemBuild executado sobre um SO Windows XP e um PC Intel Celeron 895 MHz (implementação 1 anterior, Figura 4.2a) e comparados aos gerados no simulador gerado pelo ambiente MatrixX/AutoCode, compilado para o SOTR RTEMS executado sobre o SIS, que executa sobre o sistema operacional Linux com processador Pentium D 820 com 2.8GHZ (ERC32 Multi-processado) (implementação 5 anterior, Figura 4.2c), ambos feitos por Amorim III (2009).

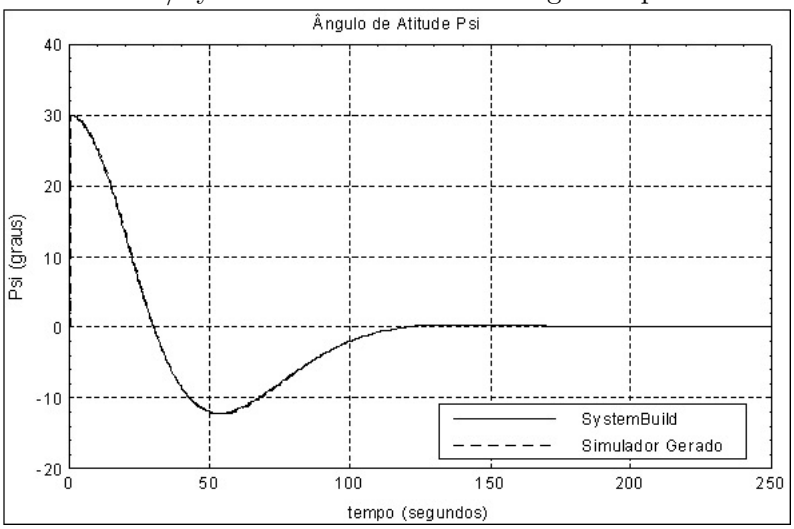
- Gerados no ambiente MatrixX/SystemBuild instalado sobre um SO Win-

dows XP instalado sobre uma máquina virtual VirtualBox por sua vez instalada em PC com SO Linux Mint com processador Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz e 16 GB de RAM (implementação 1 atual, Figura 4.3a) e comparados aos gerados no simulador gerado e compilado para o SOTR RTEMS executado sobre o processador real ERC32 (implementação 3 atual, Figura 4.3b), ambos feitos neste trabalho.

Os resultados para uma mesma variável e implementações diferentes não foram comparados em um mesmo gráfico, pois o autor não obteve acesso aos dados crus de simulação do trabalho de Amorim III (2009); porém, obteve acesso às figuras dos resultados de seu trabalho. As demais variáveis podem ser vistas no Apêndice A.

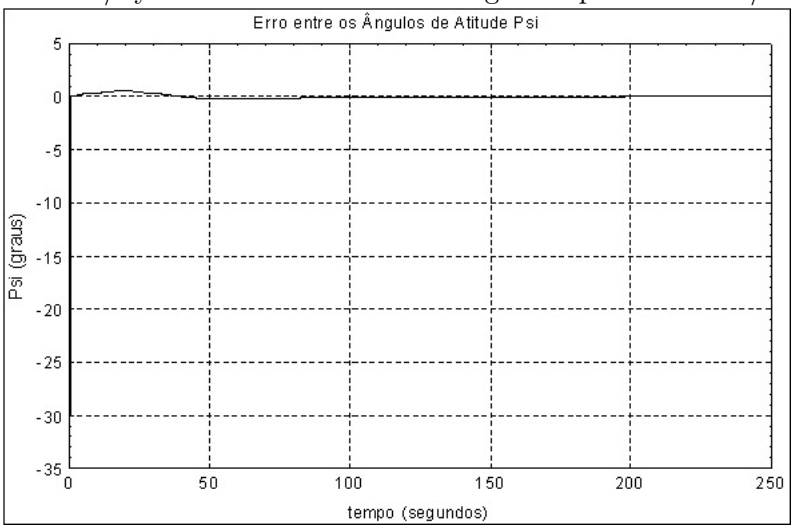
5.2.1 Ângulo de atitude ψ (Psi)

Figura 5.1 - Comparação entre os valores do ângulo ψ (Psi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



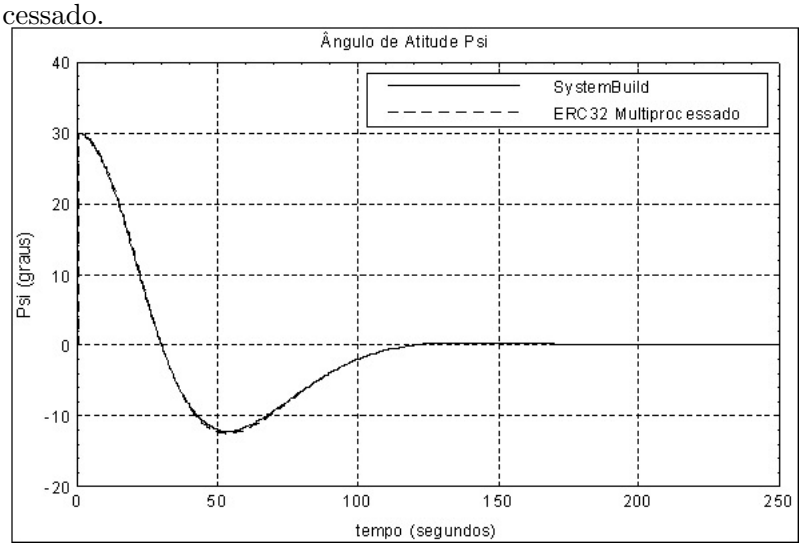
Fonte: Amorim III (2009).

Figura 5.2 - Erro entre os valores do ângulo ψ (Psi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



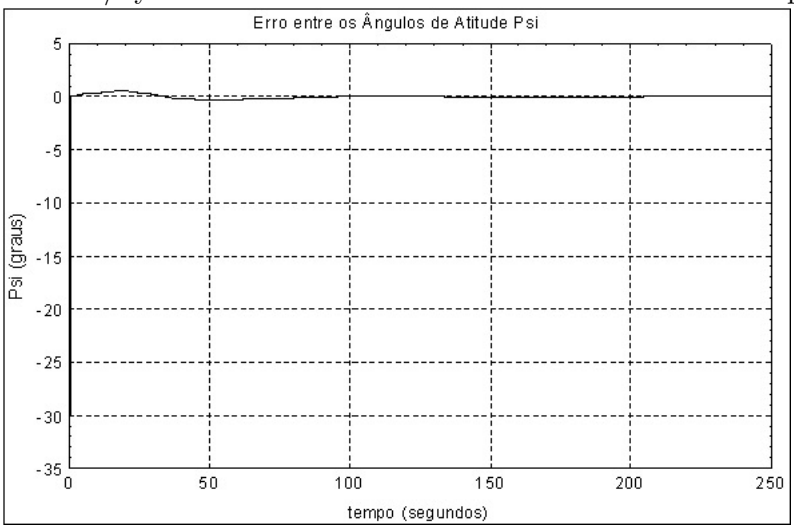
Fonte: Amorim III (2009).

Figura 5.3 - Comparação entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



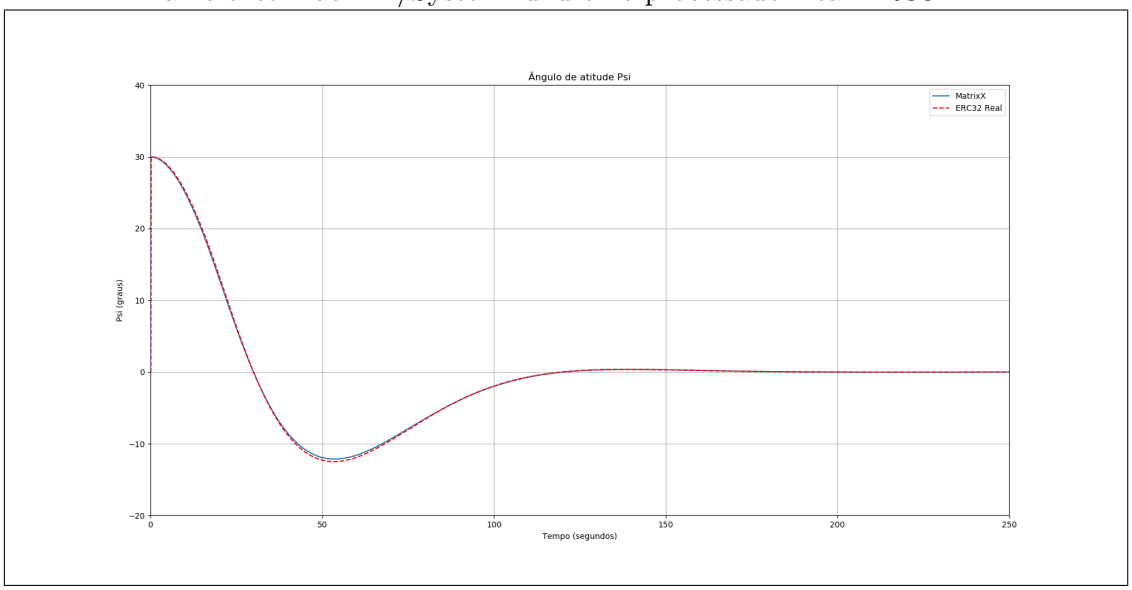
Fonte: Amorim III (2009).

Figura 5.4 - Erro entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



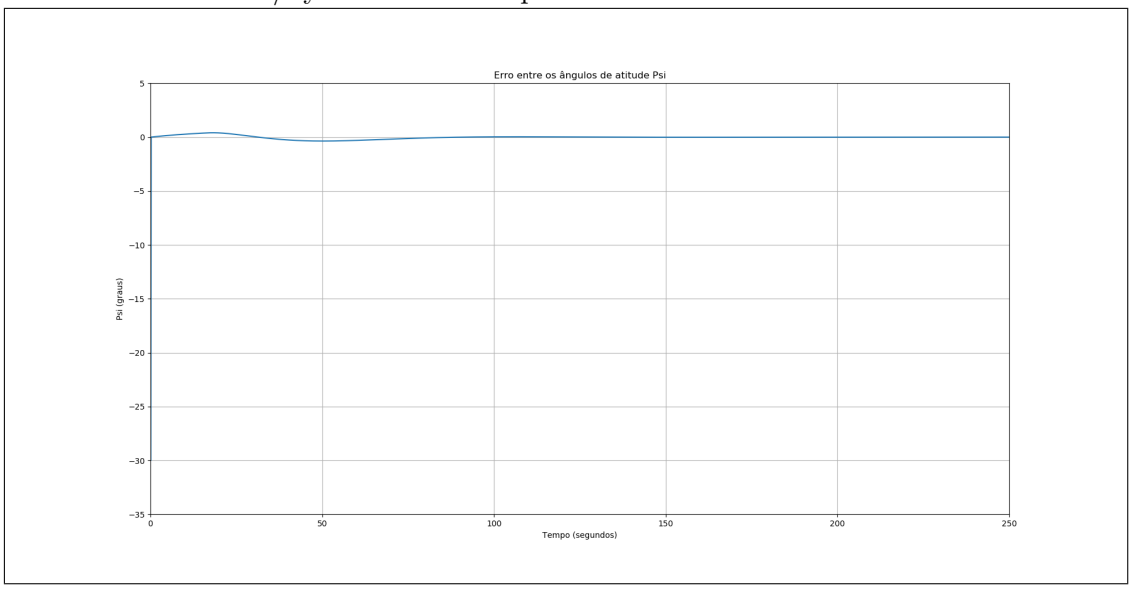
Fonte: Amorim III (2009).

Figura 5.5 - Comparação entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

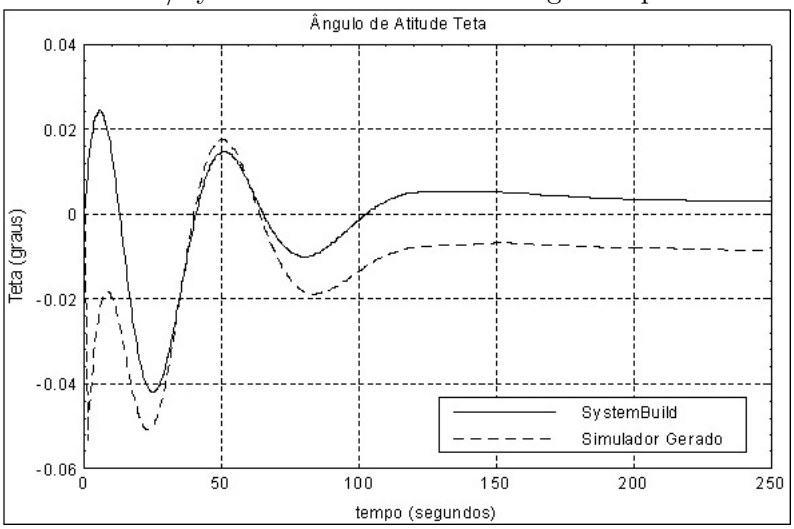
Figura 5.6 - Erro entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

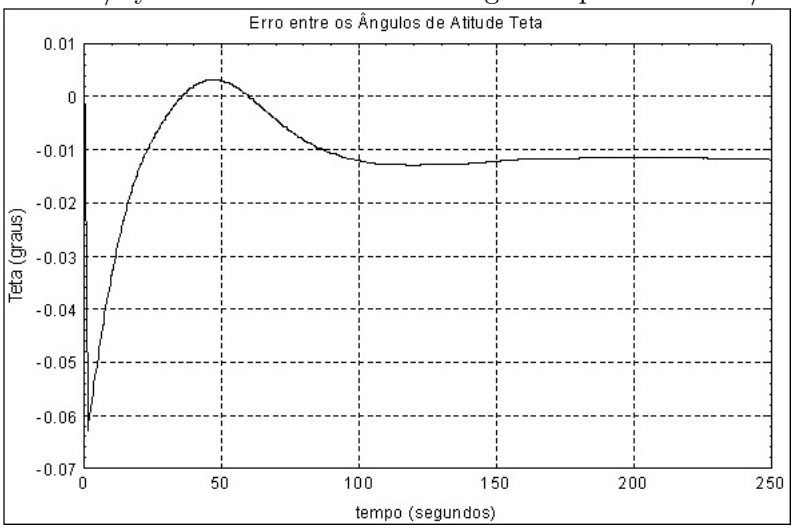
5.2.2 Ângulo de atitude θ (Teta)

Figura 5.7 - Comparação entre os valores do ângulo θ (Teta) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



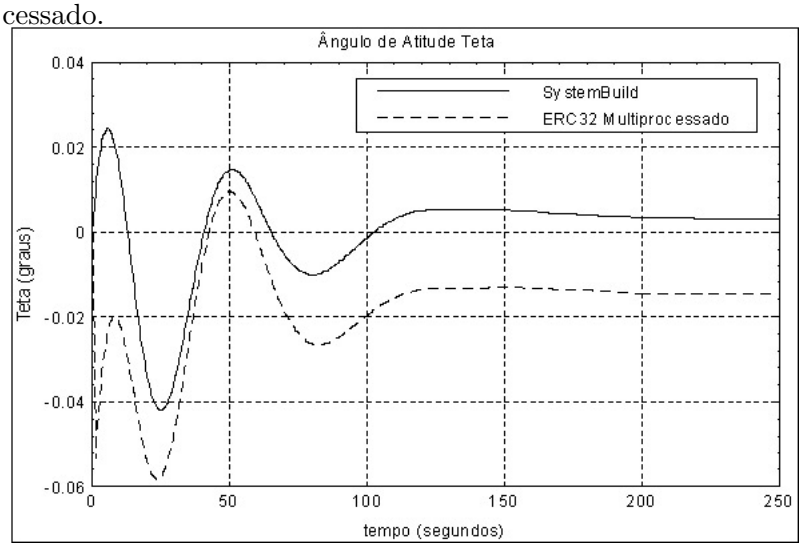
Fonte: Amorim III (2009).

Figura 5.8 - Erro entre os valores do ângulo θ (Teta) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



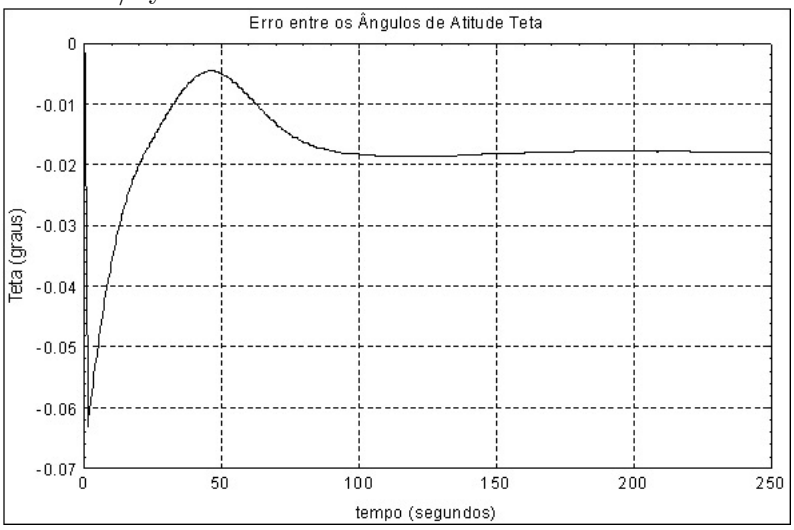
Fonte: Amorim III (2009).

Figura 5.9 - Comparação entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



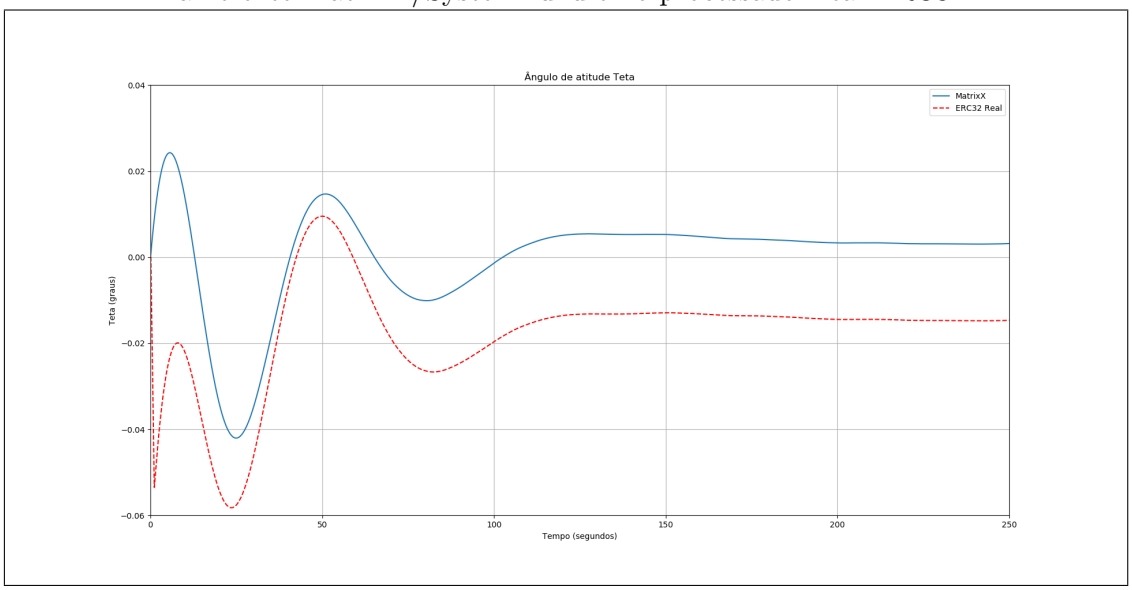
Fonte: Amorim III (2009).

Figura 5.10 - Erro entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



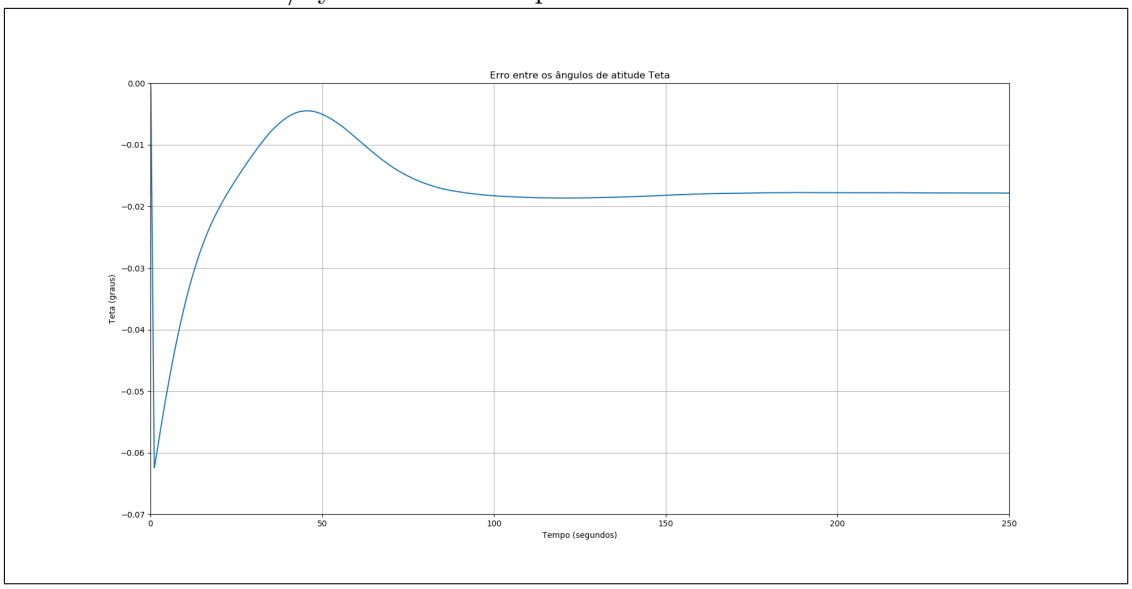
Fonte: Amorim III (2009).

Figura 5.11 - Comparação entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

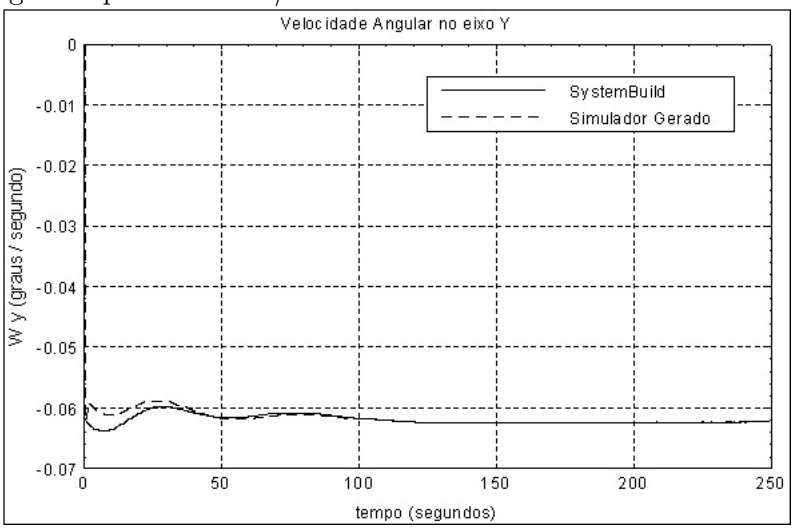
Figura 5.12 - Erro entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

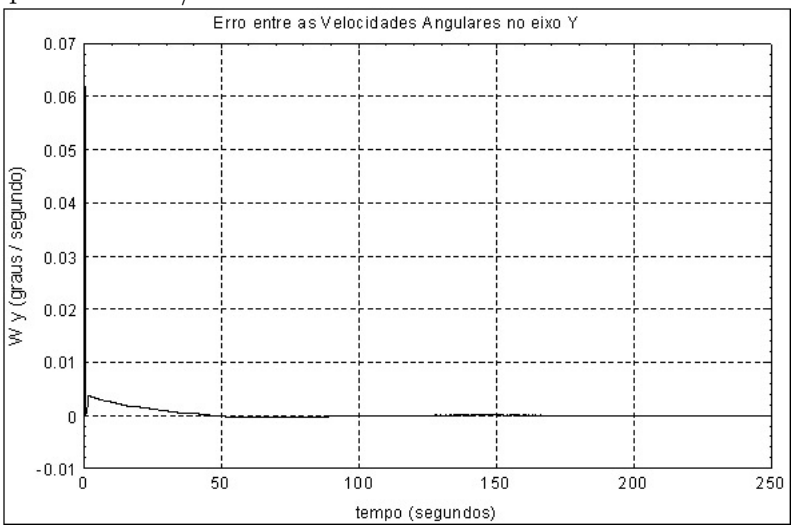
5.2.3 Velocidade angular no eixo Y (Wy)

Figura 5.13 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



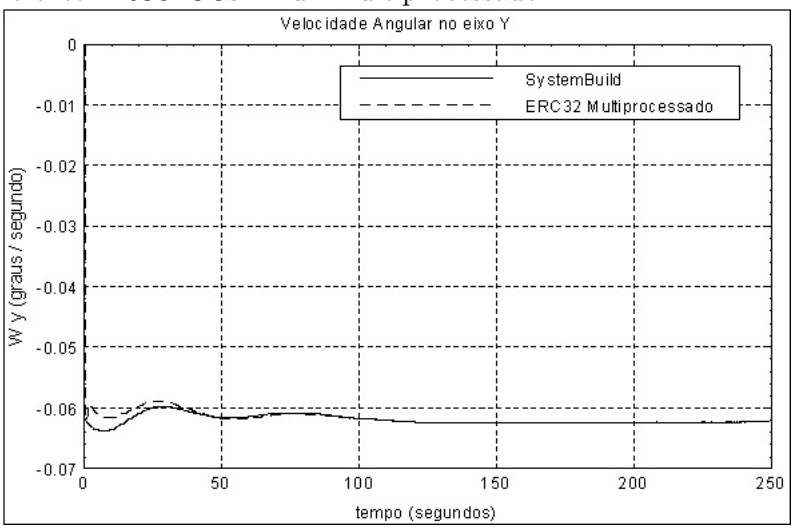
Fonte: Amorim III (2009).

Figura 5.14 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



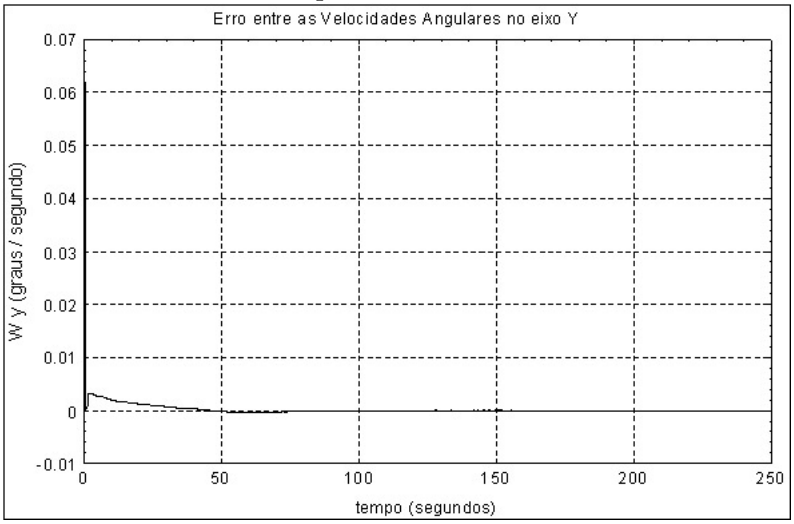
Fonte: Amorim III (2009).

Figura 5.15 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



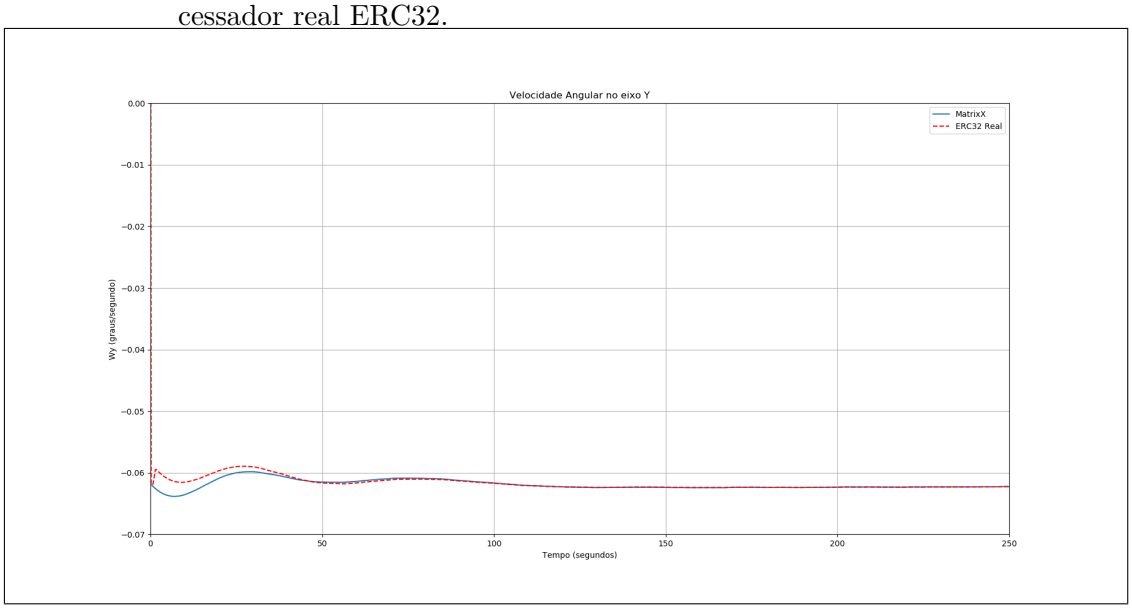
Fonte: Amorim III (2009).

Figura 5.16 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



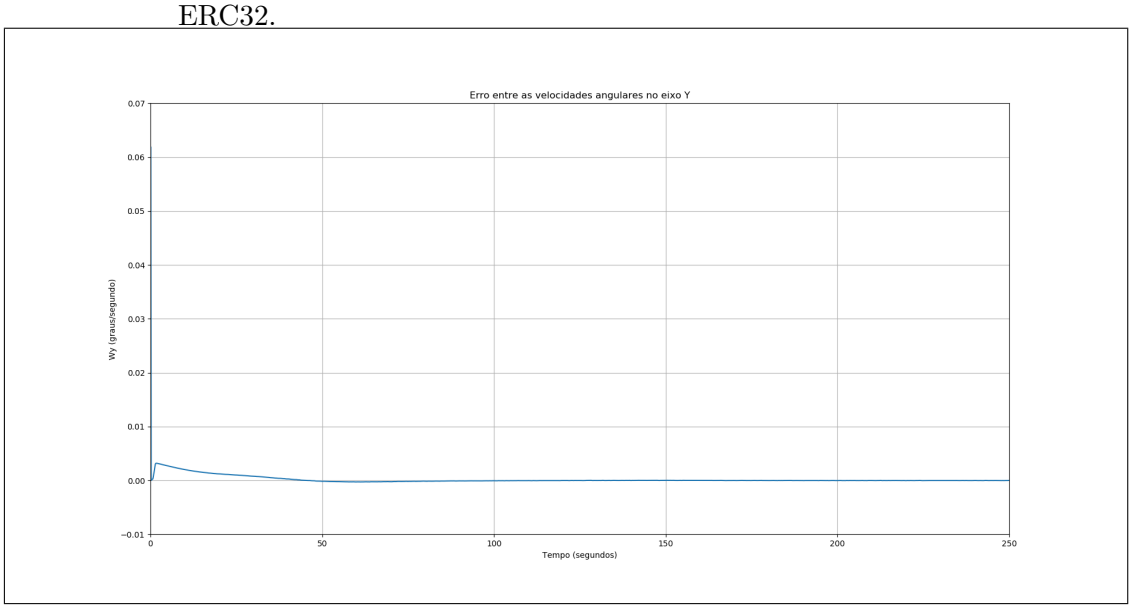
Fonte: Amorim III (2009).

Figura 5.17 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

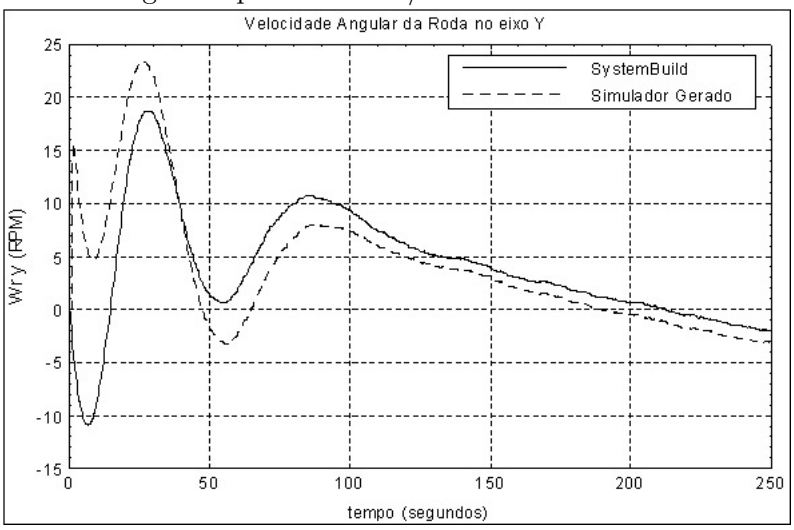
Figura 5.18 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

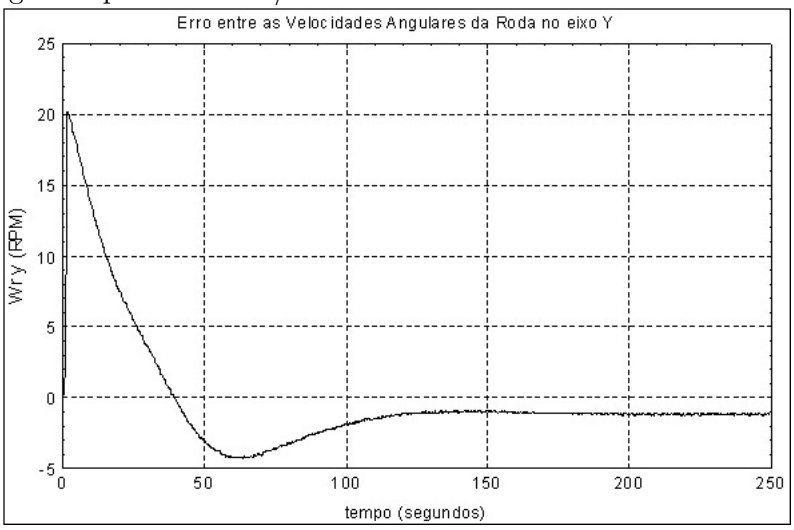
5.2.4 Velocidade angular da roda de reação no eixo Y (Wry)

Figura 5.19 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



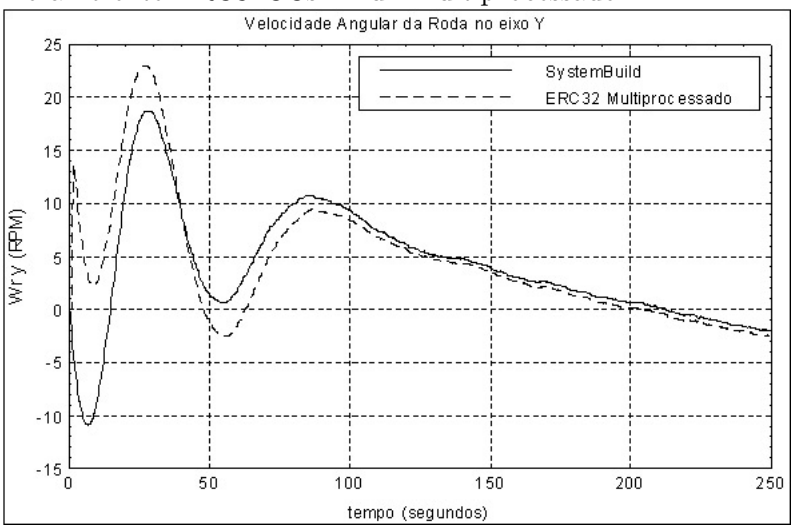
Fonte: Amorim III (2009).

Figura 5.20 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



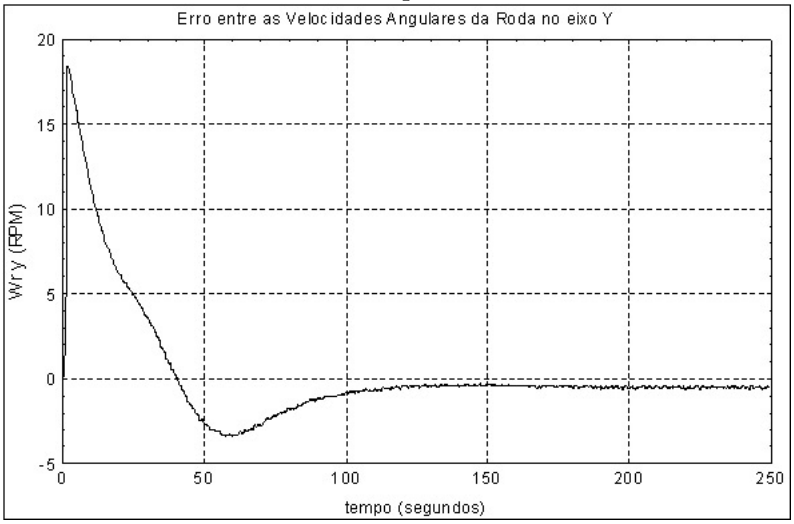
Fonte: Amorim III (2009).

Figura 5.21 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



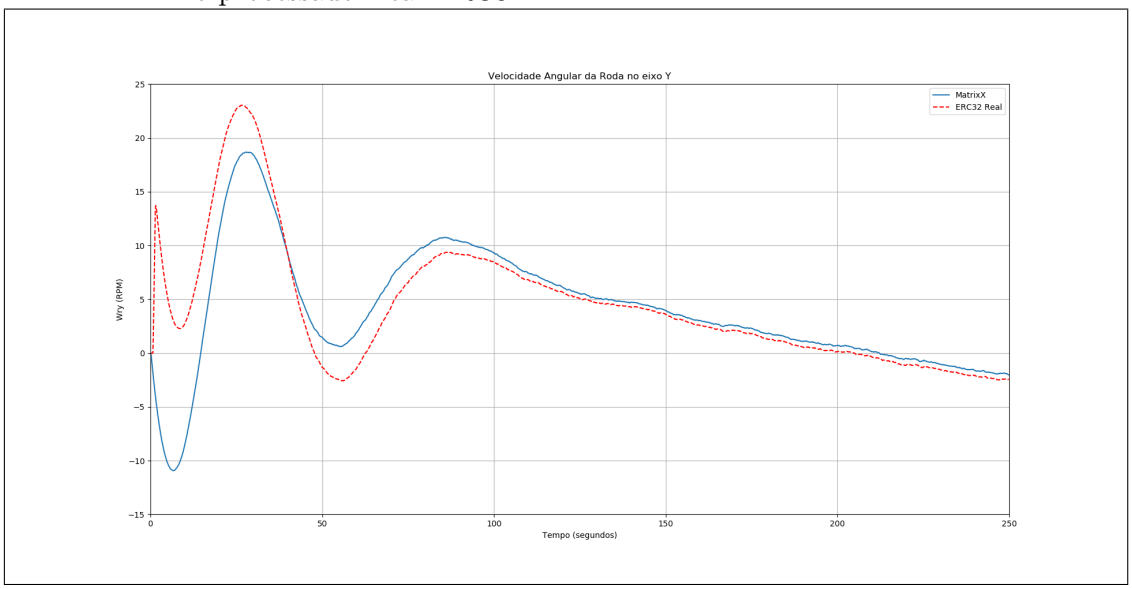
Fonte: Amorim III (2009).

Figura 5.22 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCs-Linux Multiprocessado.



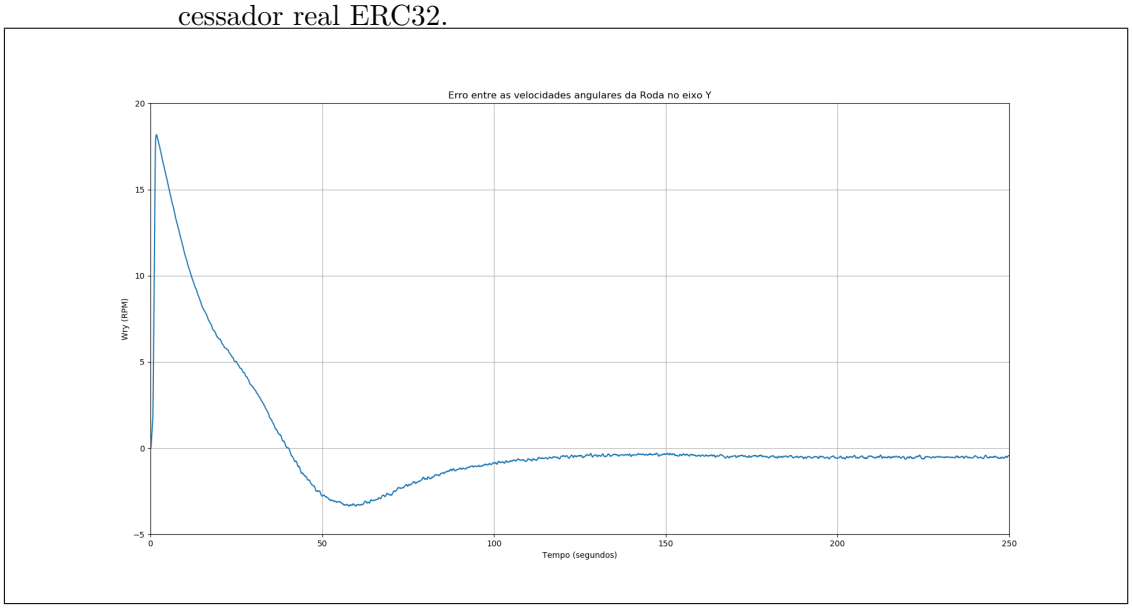
Fonte: Amorim III (2009).

Figura 5.23 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

Figura 5.24 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

5.3 Análise

Os gráficos aqui apresentados das simulações realizadas com o ERC32 real demonstram extrema semelhança com os gráficos das simulações geradas pela implementação de Amorim III (2009). Esta semelhança se preserva em todas as variáveis e pode ser obervada nos demais gráficos que se encontram no Apêndice A. Nota-se que o comportamento do ERC32 Real é graficamente idêntico ao do ERC32 simulado e que o erro entre o ERC32 real e o MatrixX/SystemBuild e o erro entre o ERC32 simulado e o MatrixX/SystemBuild se comportam da mesma forma. No trabalho de Amorim III (2009), as pequenas diferenças nos primeiros instantes de tempo independentes da implementação, ocorrem devido à inicialização do sistema. Neste processo a ferramenta MatrixX/SystemBuild e a ferramenta MatrixX/AutoCode atribuem um valor inicial muito próximo de zero (zero numérico) para as variáveis. Porém, apesar dos valores serem muito pequenos, eles podem diferir: por exemplo, um pode ser negativo e outro positivo, o que implica uma trajetória inicial diferente. É importante ressaltar que o Controlador de Atitude possui ação integral, o que faz com que haja uma diferença de valor significativa maior no regime transitório. Isto ocorre porque qualquer diferença numérica nos primeiros instantes é integrada junto com as variáveis. Isto pode ser observado claramente nos gráficos das variáveis ân-

gulo de atitude Teta (θ) da Subseção 5.2.2 e velocidade angular da roda de reação no eixo Y da Subseção 5.2.4.

Nota-se que o ângulo de atitude Psi (ψ) (Subseção 5.2.1), principal da manobra, atingiu regime permanente na faixa dos 180 segundos e se manteve praticamente em 0°.

A Maior diferença em regime permanente pode ser observada nos gráficos do ângulo de atitude Teta (θ) da Subseção 5.2.2. Esta diferença já existia no trabalho de Amorim III (2009) e permanece neste trabalho. Porém, este erro de, no máximo, 0,02°, é menor que a tolerância aceita pelo requisito de apontamento da manobra que é de 0,05°.

Nos gráficos da velocidade angular no eixo Y (Wy) da Subseção 5.2.3 é possível notar uma deriva de aproximadamnte -0.062°/s. Segundo Amorim III (2009), esta deriva ocorre devido ao movimento orbital.

6 CONCLUSÕES

Este trabalho foi uma continuação direta do trabalho de Amorim III (2009), o qual apresentou cinco implementações distintas para simulação em tempo real de um Sistema de Controle de Atitude (SCA) de um modelo similar ao da Plataforma MultiMissão (PMM) de 2001 em seu Modo Nominal de operação. As implementações realizadas por ele foram puramente computacionais, diferenciando-se apenas na plataforma em que foram executadas. O presente trabalho estendeu o trabalho de Amorim III (2009) por meio da migração do SCA para um processador real ERC32. A implementação foi realizada através da técnica *Processor-In-The-Loop*.

No trabalho de Amorim III (2009), as simulações em tempo real realizadas no MatrixX/SystemBuild, como as realizadas pelas implementações em ambiente computacionais diferentes, atendem aos requisitos de desempenho de um SCA para um modelo similar ao da PMM de 2001 em Modo Nominal de operação. Os gráficos da Seção 5.2 e do Apêndice A demonstram que o comportamento dinâmico da implementação da simulação com processador ERC32 é praticamente igual ao comportamento das implementações de Amorim III (2009); logo, atendem aos requisistos de desempenho. Os simuladores, tanto no trabalho de Amorim III (2009) quanto no presente trabalho, foram obtidos por meio de geração automática de códigos pelo MatrixX/AutoCode a partir dos modelos em blocos do MatrixX/SystemBuild.

No trabalho anterior nota-se que as três últimas implementações (3ª, 4ª e 5ª) apresentam comportamento extremamente semelhantes entre si porém diferem em relação a simulação realizada no MatrixX/SystemBuild (1ª implementação) e MatrixX/AutoCode(2ª implementação). Segundo Amorim III (2009), isto informa que a migração de uma simulação em tempo virtual gerada pelo MatrixX/AutoCode para um simulador em tempo real introduz mais diferenças do que as implementações em ambientes computacionais distintos. Essa diferença permanece neste trabalho e é confirmada pela acentuada semelhança dos resultados de simulação da implementação atual em processador ERC32 com as distintas implementações anteriores realizadas por Amorim III (2009).

O presente trabalho conseguiu realizar uma simulação de um SCA em um processador ERC32 real em malha fechada por meio da abordagem temporal executiva cíclica. O requisito de período de amostragem foi atendido pelo simulador; portanto, a simulação atende satisfatoriamente requisitos de tempo real. O sistema de comunicação desenvolvido para realizar o envio e recebimentos de dados de forma ordenada entre o computador PC (Mundo externo) e o processador ERC32 (Con-

trolador de Atitude) funciona de forma satisfatória. Tal sistema pode ser adaptado para a comunicação em malha fechada com outros processadores candidatos para satélites.

É importante notar que a implementação realizada neste trabalho validou logicamente os códigos gerados pelo MatrixX/AutoCode oriundos dos modelos em MatrixX/SystemBuild utilizados por Amorim III (2009). Isto significa que o codigo do Controlador de Atitude (SCA) executou sobre a arquitetura do processador ERC32 de maneira satisfatória; em outros termos, ele não apresentou erros lógicos nas operações de cálculo. Isto pode ser confirmado pelos resultados semelhantes aos das implementações realizadas por Amorim III (2009).

Infelizmente, o ambiente computacional MatrixX utilizado neste trabalho está defasado e o suporte para o mesmo foi descontinuado pela National Instruments em 2018 (NI FORUM, 2020). É possível que ambientes computacionais atuais consigam gerar códigos cuja adaptação a uma determinada arquitetura ou processador específico seja mais facilmente realizada.

Duas grandes dificuldades atrasaram o desenvolvimento do presente trabalho. A primeira foi a pandemia de Covid-19, que devidos às restrições impostas pela mesma dificultou acesso ao laboratório onde a implementação principal deste trabalho foi desenvolvida. A segunda foi uma infiltração que ocorreu no mesmo laboratório que, consequentemente, causou danos aos equipamentos utilizados para realizar este trabalho.

Este trabalho conseguiu atender a duas sugestões para futuros estudos do trabalho de Amorim III (2009), a saber:

- A utilização de interfaces com hardware mais fiéis à realidade;
- Aumentar novamente o grau de fidelidade destas simulações por meio da inclusão de um processador físico na malha, semelhante ao que será utilizado para o vôo.

Sugere-se para trabalhos futuros:

- Os pontos não atendidos do trabalho de Amorim III (2009);
- Utilizar e aperfeiçoar abordagens de temporização diferentes da utilizada no presente trabalho;

- Adaptar a parte de comunicação do sistema de simulação desenvolvido neste trabalho para o teste de outros códigos;
- Adaptar o sistema de simulação desenvolvido neste trabalho para outros processadores, especialmente os da família LEON;
- Utilizar um ambiente de modelagem moderno especialmente um com geração de código automática altamente personalizável;
- Utilizar outro protocolo de comunicação entre as partes como por exemplo: MIL-STD-1553 ou Time-Triggered Protocol (TTP);
- Administrar a base de tempo do simulador pelo Controlador de Atitude.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORIM III, F. C. **Simulação paralela de um controle discreto de atitude, com dois processadores e um sistema operacional de tempo real, para a Plataforma MultiMissão**. 164 p. 213. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2009. Acesso em: 2006. xii, xxvi, 1, 2, 6, 11, 14, 15, 16, 17, 19, 20, 21, 24, 25, 26, 28, 30, 31, 33, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 82, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 113, 114, 116, 117, 119, 120, 122, 123, 125, 126, 128, 129, 131, 132, 134, 135

ARANTES JÚNIOR, G. **Estudo comparativo de técnicas de controle de atitude em 3 eixos para satélites artificiais**. 187 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2005. 29

ARDAKANI, H. A.; BRIDGES, T. J. **Review of the 3-2-1 euler angles: a yaw-pitch-roll sequence**. 2021. Disponível em: <http://personal.maths.surrey.ac.uk/T.Bridges/SLOSH/3-2-1-Eulerangles.pdf>. Acesso em: 24 abr. 2021. 35, 36

BARBOSA, D. S. **Simulação e implementação digital em tempo real do controle autônomo da atitude de satélites estabilizados por rotação via bobinas magnéticas**. 232 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2000. 15, 20

BARRETO, J. M. A. B.; SOUZA, M.; AMORIM III, F. C. A. **Relatório final de projeto de iniciação científica (PIBIC/CNPq/INPE)**. São José dos Campos: INPE: [s.n.], 2012. 17

COBHAM GAISLER. **Processors**. 2021. Disponível em: <https://www.gaisler.com/index.php/products/processors>. Acesso em: 28 jan. 2021. 12

CURTIS, H. **Orbital mechanics for engineering students**. 3. ed. Oxford, UK: Butterworth-Heinemann, 2014. 751 p. 36

DONIZETE, D.; SOUZA, M.; AMORIM III, F. C. A. **Relatório parcial de projeto de iniciação científica (PIBIC/CNPq/INPE)**. São José dos Campos, INPE: [s.n.], 2011. 17

EUROPEAN SPACE AGENCY - ESA. **Advanced star tracker**. 2020. Disponível em: <https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Alphasat/Advanced_Star_Tracker>. Acesso em: 15 out. 2020. 40

_____. **RTEMS**. 2021. Disponível em: <https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Software_Systems_Engineering/RTEMS>. Acesso em: 27 jan. 2021. 11

FARINES, J. M.; FRAGA, J. S.; OLIVEIRA, R. **Sistemas de tempo real**. Florianópolis: Departamento de Automação e Sistemas, 2000. 208. 9, 15

FRANKLIN, G. F.; POWELL, J. D. **Digital control of dynamic systems**. 2. ed. Massachusetts, USA: Addison-Wesley, 1981. 335 p. 41

GOBATO, M. F. **Controles monovariáveis e multivariáveis aplicados a sistemas aeroespaciais fracamente ou fortemente acoplados**. 388 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006. 1, 16, 19, 20, 21, 28, 32, 40, 41, 42, 44, 45, 49

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS - INPE. **Amazonia1**. 2020. Disponível em: <http://www.inpe.br/amazonia1/sobre_satelite/pmm.php>. Acesso em: 9 out. 2020. 13, 14, 19

_____. _____. 2021. Disponível em: <http://www.inpe.br/noticias/noticia.php?Cod_Noticia=5706>. Acesso em: 24 abr. 2021. 19

KALE, M.; GHATWAI, N.; REPUDI, S. **Processor-In-Loop simulation: embedded software verification and validation in model based development**. 2020. Disponível em: <https://www.design-reuse.com/articles/42548/embedded-software-verification-validation-in-model-based-development.html>. Acesso em: 2 fev. 2020. 7

KAPLAN, M. H. **Modern spacecraft dynamics & control**. New York, USA: John Wiley & Sons, 1976. 415 p. 29

KUGA, H.; ZANARDI, M.; SILVA, W. O filtro h-infinito estendido de segunda ordem para estimação de atitude e bias de giros. In: CONGRESSO NACIONAL DE MATEMáTICA APLICADA E COMPUTACIONAL, 15., 2014, Natal. **Proceedings...** Natal: CNMAC, 2014. Disponível em: <https://proceedings.sbmac.org.br/sbmac/article/view/561>. Acesso em: 06 fev. 2020. 7

MORAES, A. L. O. **Apreciação, simulação e implementação da arquitetura IMA distribuída (DIMA) e sua aplicação a sistemas espaciais**. 469 p. Dissertação (Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2017. 1, 17

MOREIRA, M. L. B. **Projeto e simulação de um controle discreto para a Plataforma MultiMissão e sua migração para um sistema de tempo real**. 181 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006. 1, 16, 19, 20, 21, 28, 32, 33, 40, 41, 42, 44, 45, 49

NATIONAL INSTRUMENTS (NI). **MATRIXx - AutoCode user guide**. Austin, Texas, USA: NI, abr. 2004. 21, 22, 28

NI FORUM. **MATRIXx software entering five year EOL phase - schedule and migration options**. 2020. Disponível em: <https://forums.ni.com/t5/MATRIXx/MATRIXx-Software-Entering-Five-Year-EOL-Phase-Schedule-and/td-p/3808121?profile.language=pt-br>. Acesso em: 14 out. 2020. 21, 82

POPESCU, G. **Earth Centered Inertial (ECI) coordinate system**. 2021. Disponível em: <https://www.researchgate.net/figure/Earth-Centered-Inertial-ECI-Coordinate-System_fig2_311807426>. Acesso em: 13 abr. 2021. 30

PRUDENCIO, S. V. **Simulação digital em tempo real de um sistema de controle de atitude magnético autônomo de um satélite**. 167 p. Dissertação (Mestrado em Ciência Espacial) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2000. 15, 20

RTEMS. **RTEMS org**. 2020. Disponível em: <https://www.rtems.org/>. Acesso em: 6 fev. 2020. 10, 11

SAAB ERICSON. **32-Bit Microprocessor and computer development programme**. Goteborg, Sweden, maio 1997. 102 p. 11

SANTOS, P. **Uma apreciação do concentrador de dados remoto (RDC) em uma arquitetura IMA Distribuída (DIMA) aplicado ao controle de atitude da PMM em modo normal**. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2020. 1, 17

SOUZA, M. L. O. **Estudo e desenvolvimento de um sistema de controle de atitude ativo em três eixos para satélites artificiais usando atuadores pneumáticos a gás frio e volantes a reação**. 368 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1981. 1, 44

SOUZA, M. L. O.; TRIVELATO, G. C. Simulators and simulations - their characteristics and applications to the simulation and control of aerospace vehicles. In: CONGRESSO SAE BRASIL, 12., 2003, São Paulo. **Proceedings...** São Paulo: SAE, 2003. 5, 6, 15

STANKOVIC, J. Misconceptions about real time computing. **IEEE Transactions on Computers**, v. 21, n. 10, p. 10–19, 1988. 8, 15

TAGAWA, G. B. S. **Estudo de um controlador de atitude em um simulador de Aviônica Modular Integrada (IMA) aplicado ao satélite Amazônia- 1**. 257 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2013. 1, 17

TREDINNICK, M. R. A. C. **Controle discreto da atitude de satélites artificiais com apêndices flexíveis**. 262 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 1999. 16

TREDINNICK, P. T. A. **Simulação digital, com sensores na malha, de sistemas de control de atitude de satélites artificiais**. 172 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 1999. 16

TRIVELATO, G. C. **Controle de rodas de reação através de técnicas digitais usando modelos de referência**. 209 p. Dissertação (Mestrado em Mecânica Espacial e Controle) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 1988. 15

WERTZ, J. R. **Spacecraft atitude determination and control**. Dordrecht, Holland: D. Reidel, 1978. 858 p. 28, 29, 31, 33

WIE, B. **Space vehicle dynamics and control**. Reston, Virginia, USA: AIAA Educational Series, 1998. 661 p. 33, 34

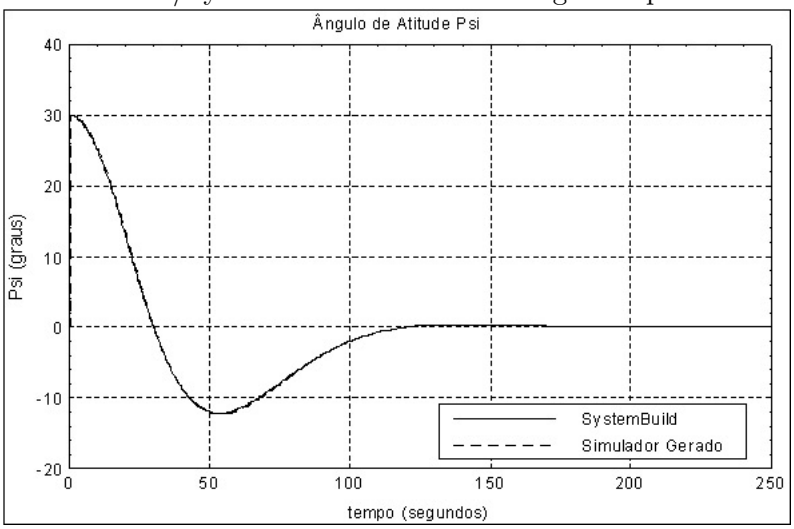
WINDRIVER. **Windriver space**. 2020. Disponível em: <https://www.windriver.com/inspace/>. Acesso em: 6 fev. 2020. 10

APÊNDICE A - GRÁFICOS DAS VARIÁVEIS

Os gráficos são apresentados segundo os mesmos critérios da Seção 5.2.

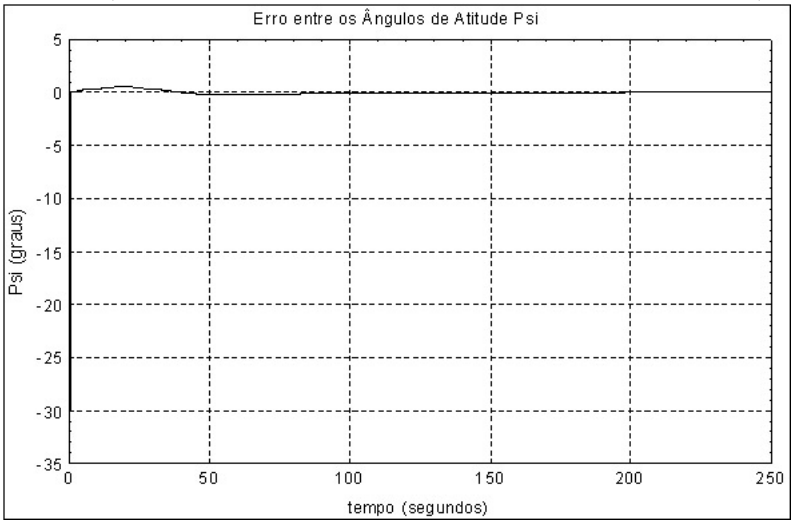
A.1 Ângulo de atitude ψ (Psi)

Figura A.1 - Comparação entre os valores do ângulo ψ (Psi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



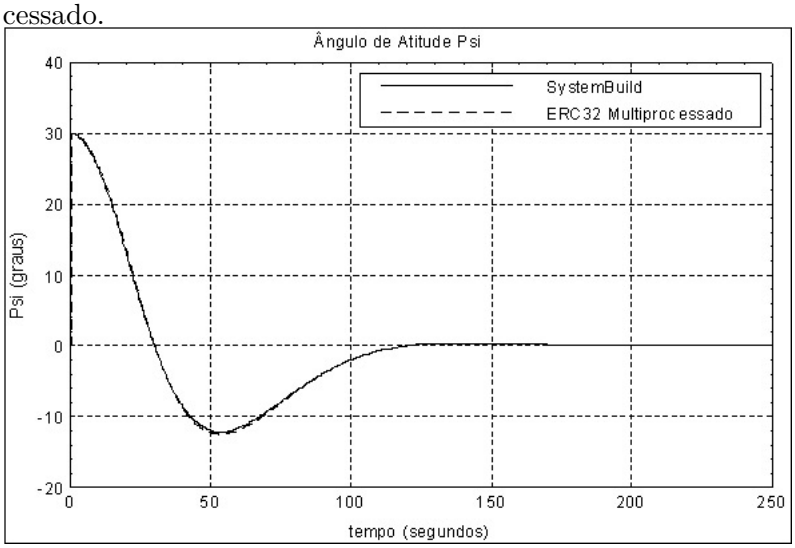
Fonte: Amorim III (2009).

Figura A.2 - Erro entre os valores do ângulo ψ (Psi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



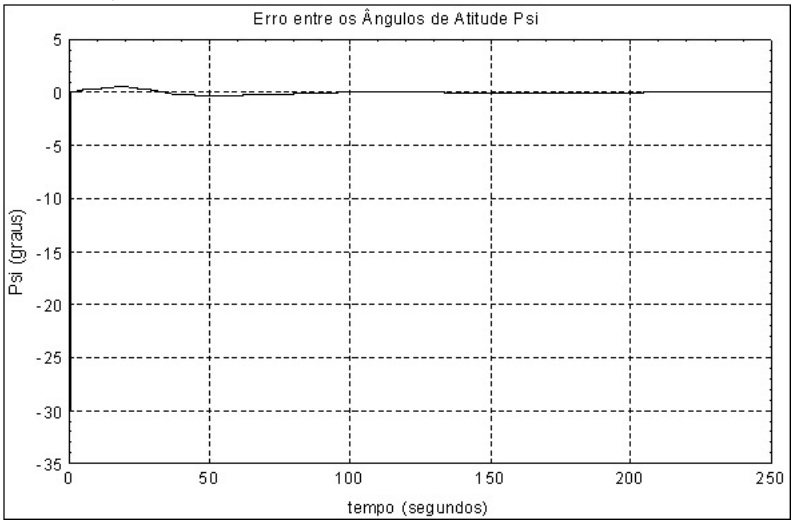
Fonte: Amorim III (2009).

Figura A.3 - Comparação entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



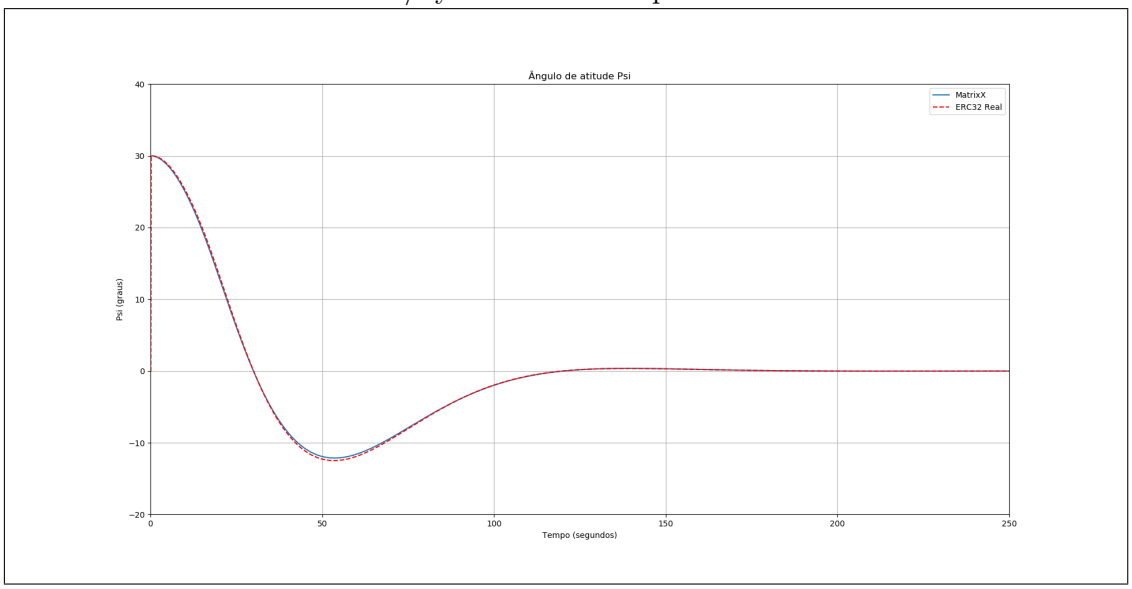
Fonte: Amorim III (2009).

Figura A.4 - Erro entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



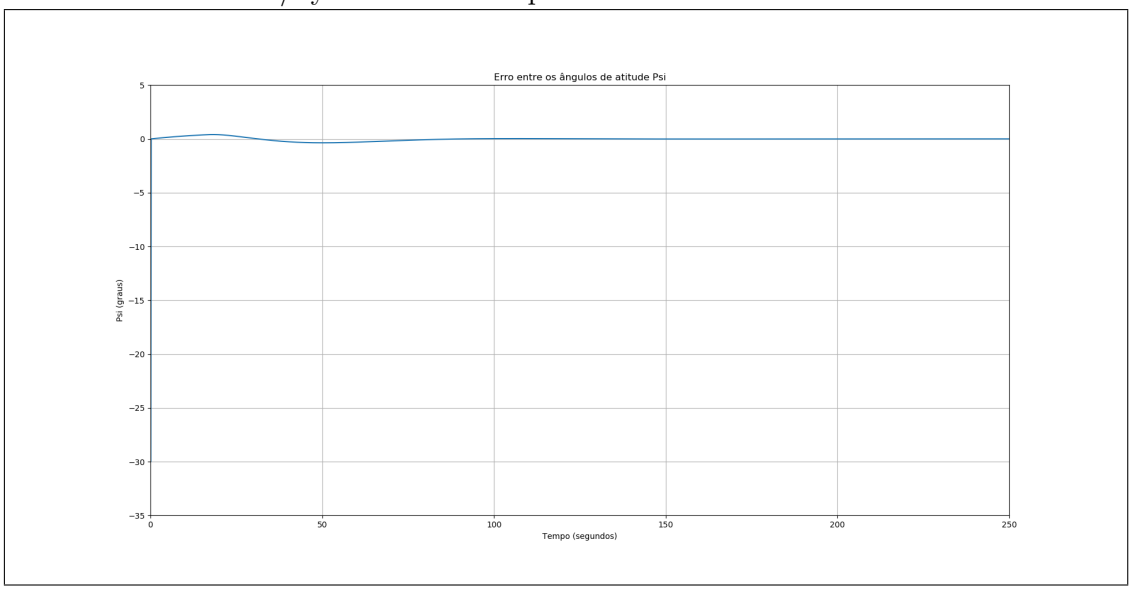
Fonte: Amorim III (2009).

Figura A.5 - Comparação entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

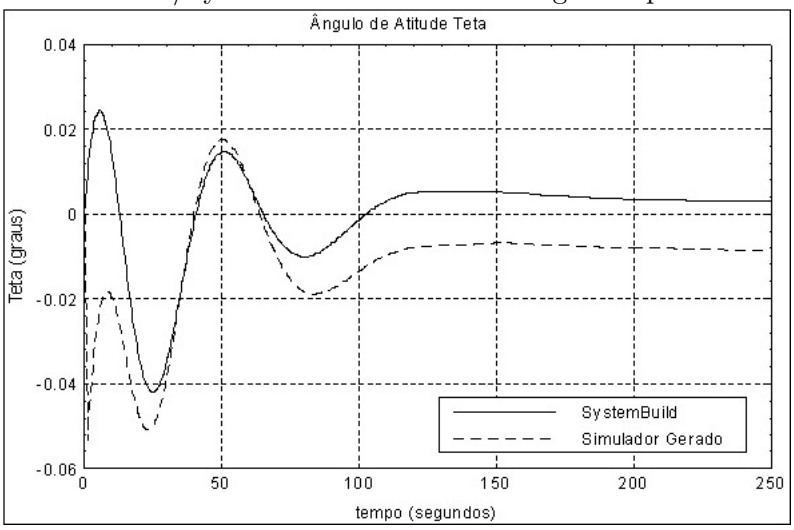
Figura A.6 - Erro entre os valores do ângulo ψ (Psi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

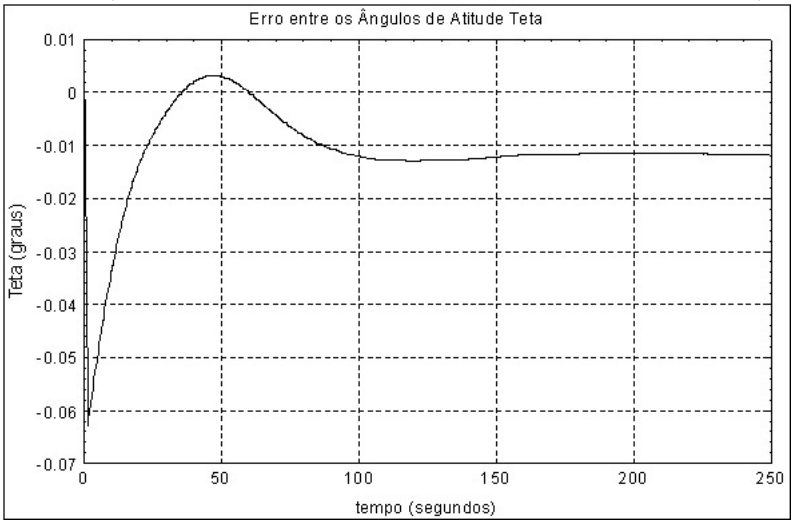
A.2 Ângulo de atitude θ (Teta)

Figura A.7 - Comparação entre os valores do ângulo θ (Teta) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



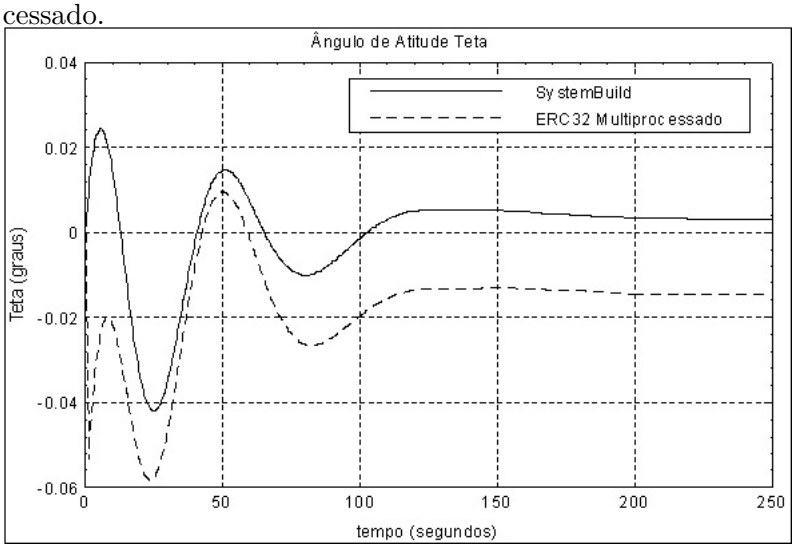
Fonte: Amorim III (2009).

Figura A.8 - Erro entre os valores do ângulo θ (Teta) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



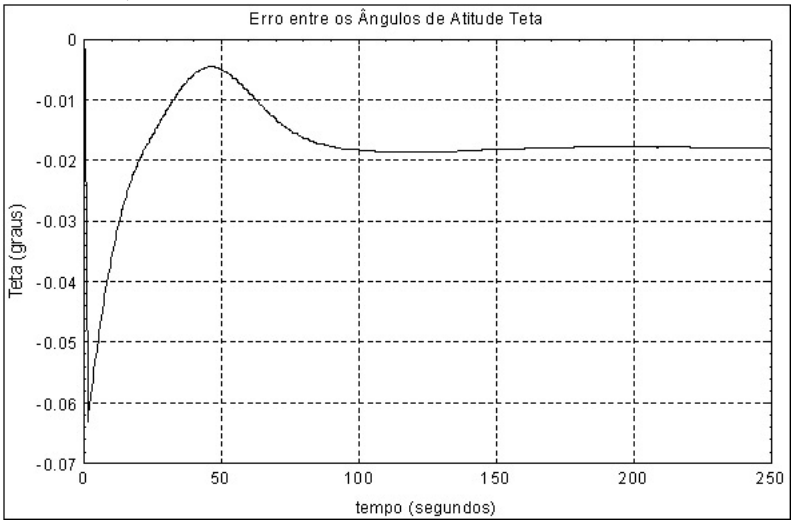
Fonte: Amorim III (2009).

Figura A.9 - Comparação entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



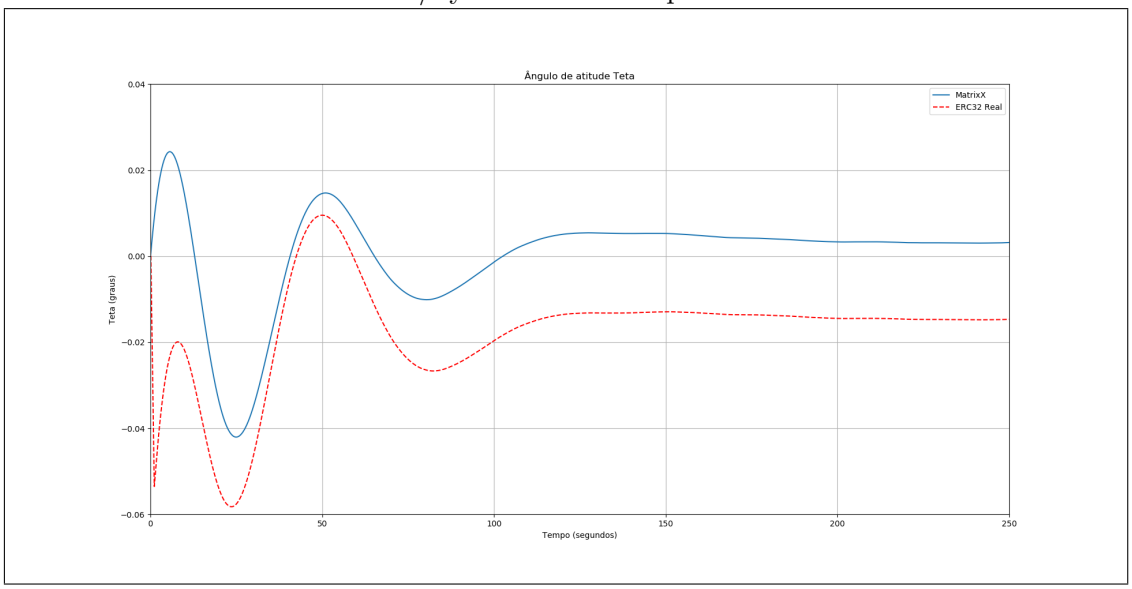
Fonte: Amorim III (2009).

Figura A.10 - Erro entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



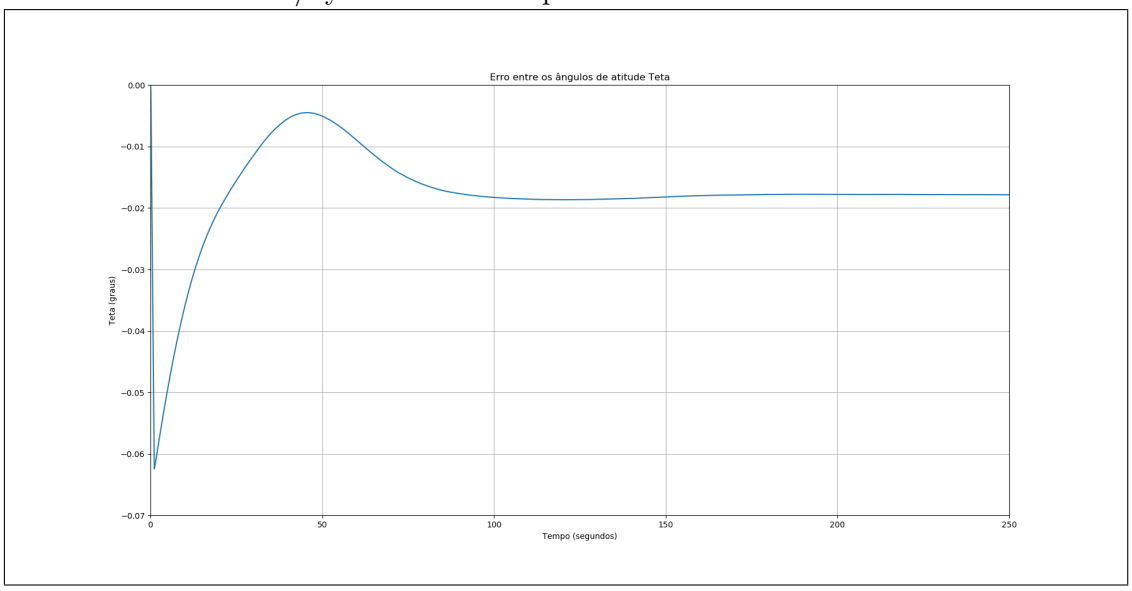
Fonte: Amorim III (2009).

Figura A.11 - Comparação entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

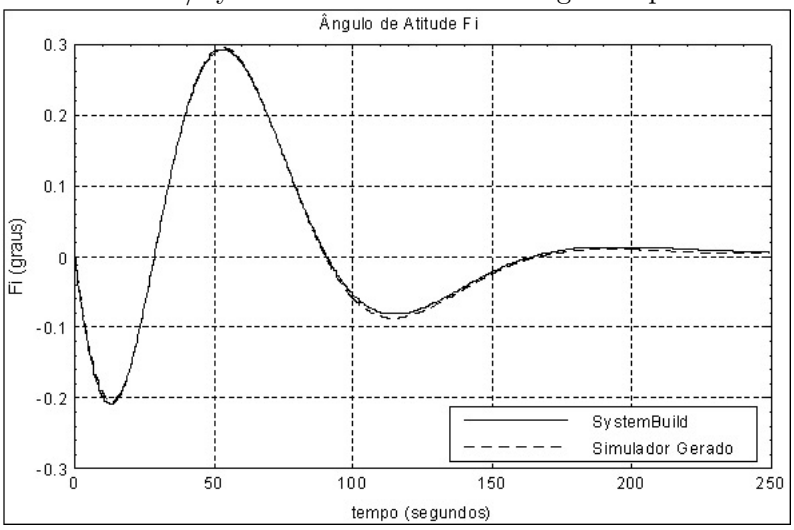
Figura A.12 - Erro entre os valores do ângulo θ (Teta) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

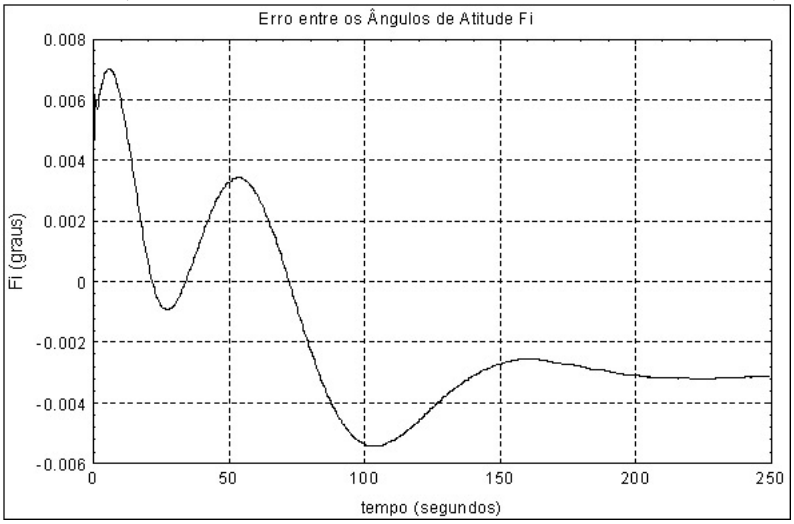
A.3 Ângulo de atitude ϕ (Fi)

Figura A.13 - Comparação entre os valores do ângulo ϕ (Fi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



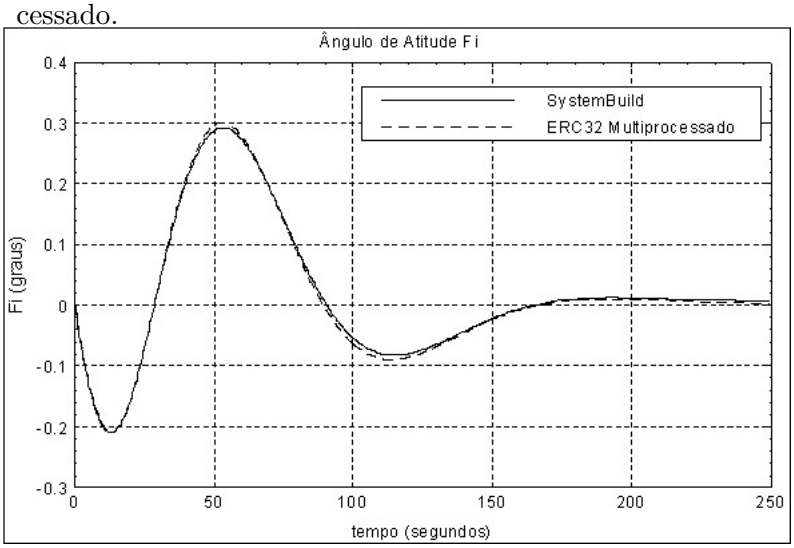
Fonte: Amorim III (2009).

Figura A.14 - Erro entre os valores do ângulo ϕ (Fi) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



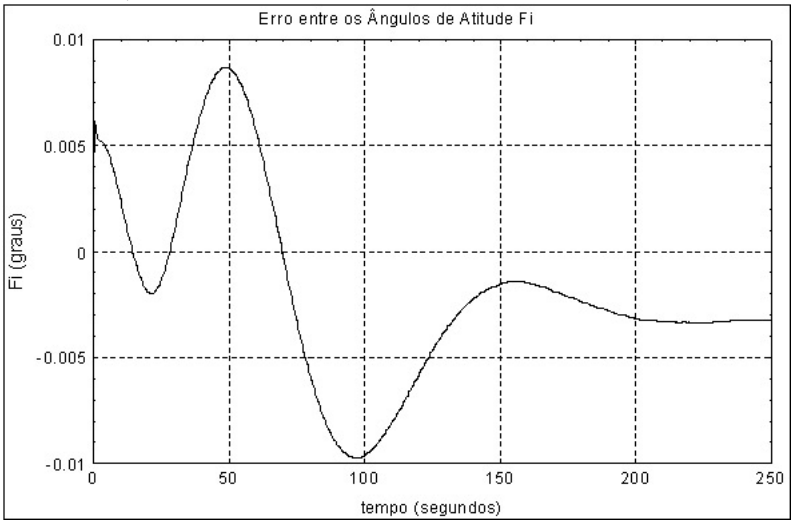
Fonte: Amorim III (2009).

Figura A.15 - Comparação entre os valores do ângulo ϕ (Fi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



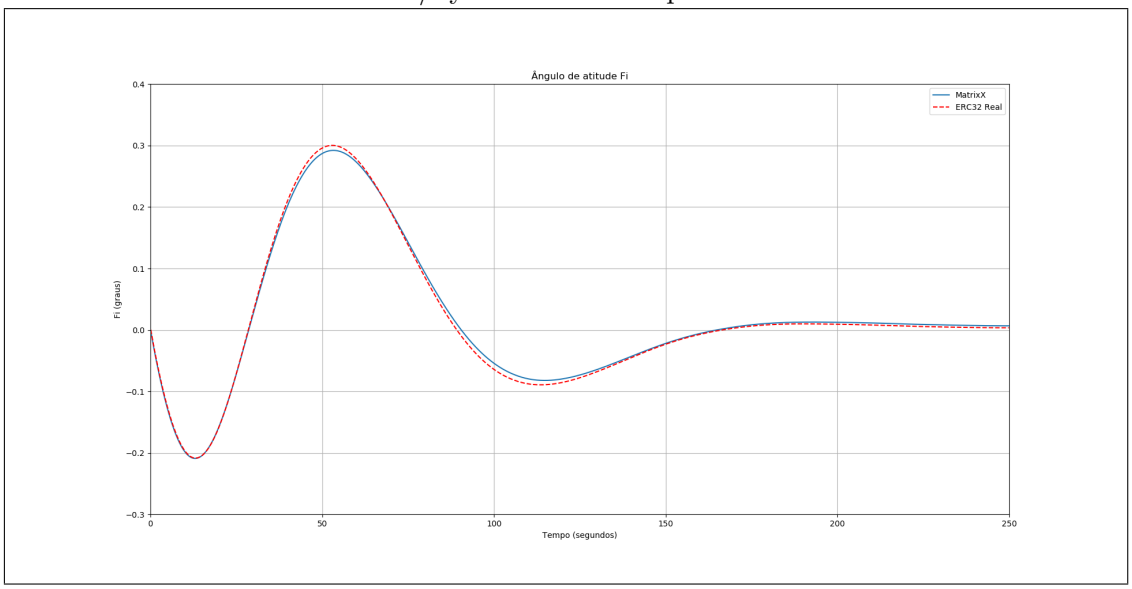
Fonte: Amorim III (2009).

Figura A.16 - Erro entre os valores do ângulo ϕ (Fi) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



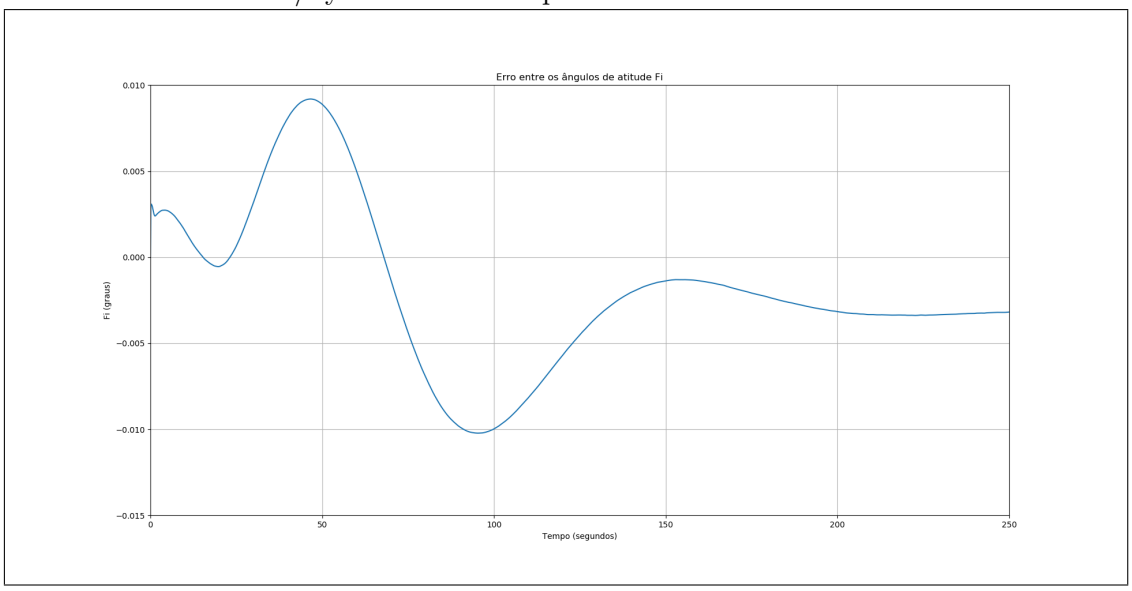
Fonte: Amorim III (2009).

Figura A.17 - Comparação entre os valores do ângulo ϕ (Fi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

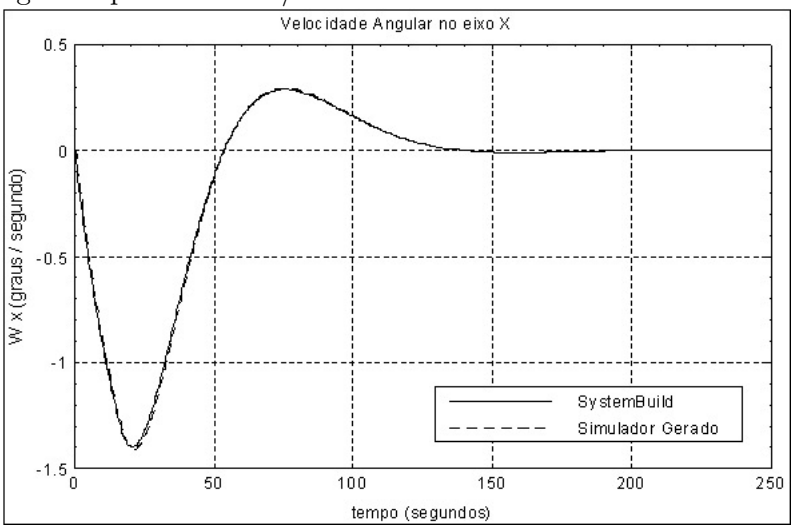
Figura A.18 - Erro entre os valores do ângulo ϕ (Fi) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

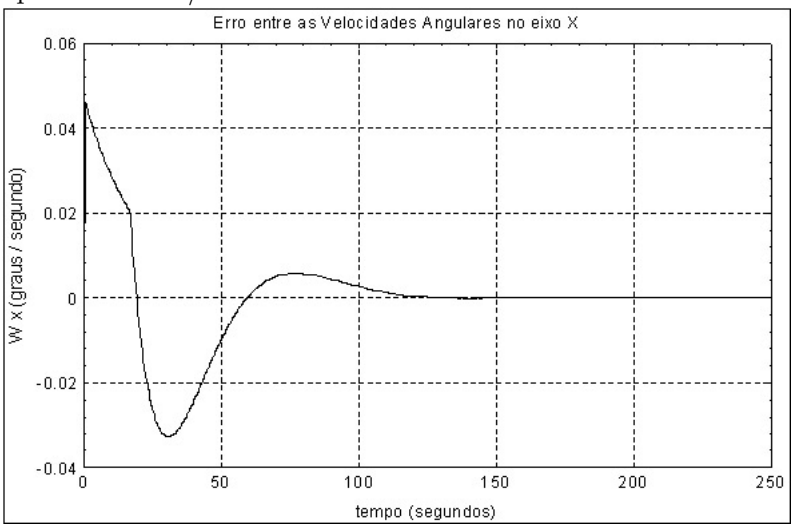
A.4 Velocidade angular no eixo X (Wx)

Figura A.19 - Comparação entre os valores da velocidade angular do satélite no eixo X (Wx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



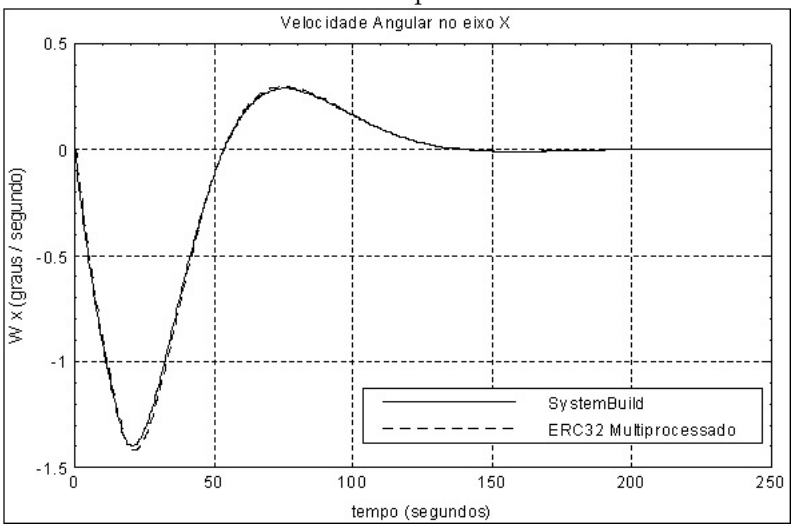
Fonte: Amorim III (2009).

Figura A.20 - Erro entre os valores da velocidade angular do satélite no eixo X (Wx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



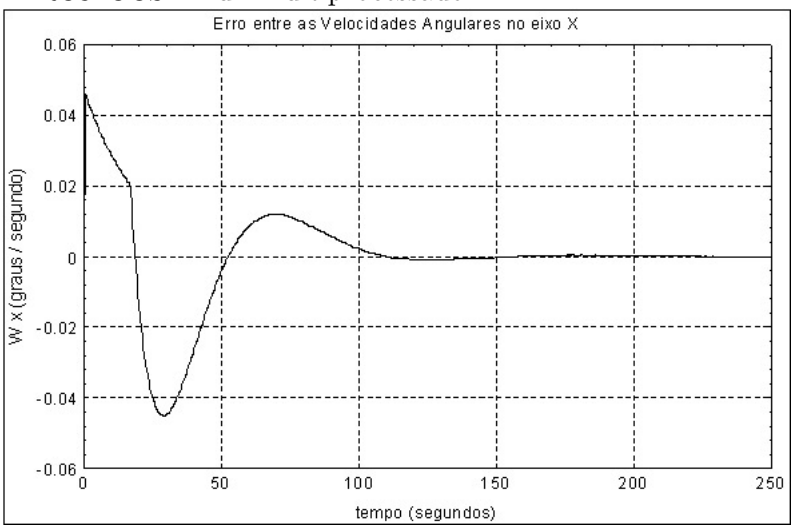
Fonte: Amorim III (2009).

Figura A.21 - Comparação entre os valores da velocidade angular do satélite no eixo X (Wx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



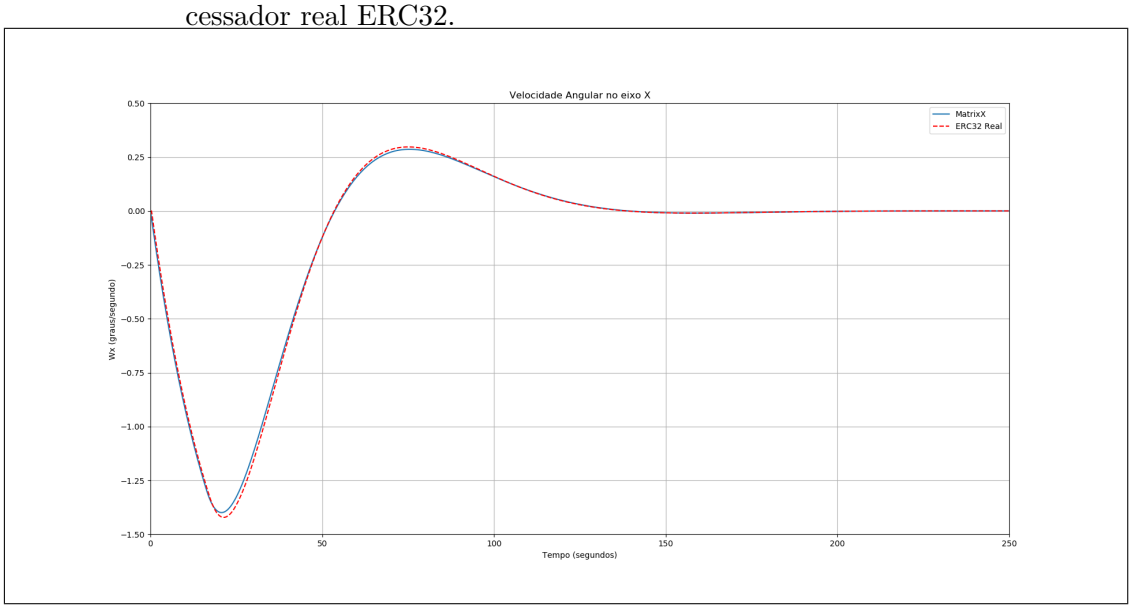
Fonte: Amorim III (2009).

Figura A.22 - Erro entre os valores da velocidade angular do satélite no eixo X (Wx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



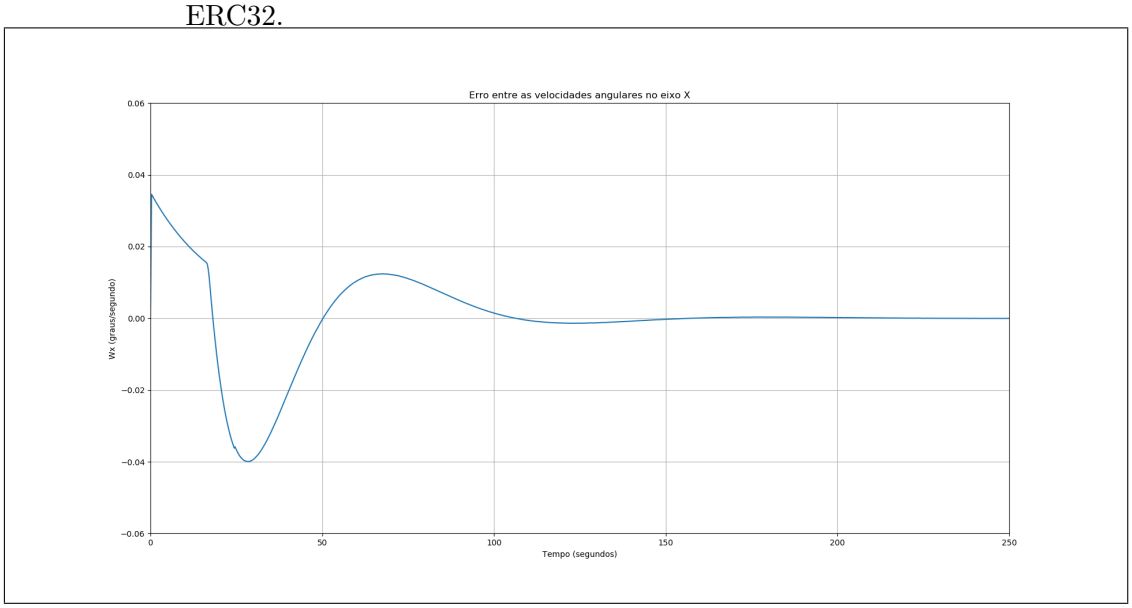
Fonte: Amorim III (2009).

Figura A.23 - Comparação entre os valores da velocidade angular do satélite no eixo X (Wx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

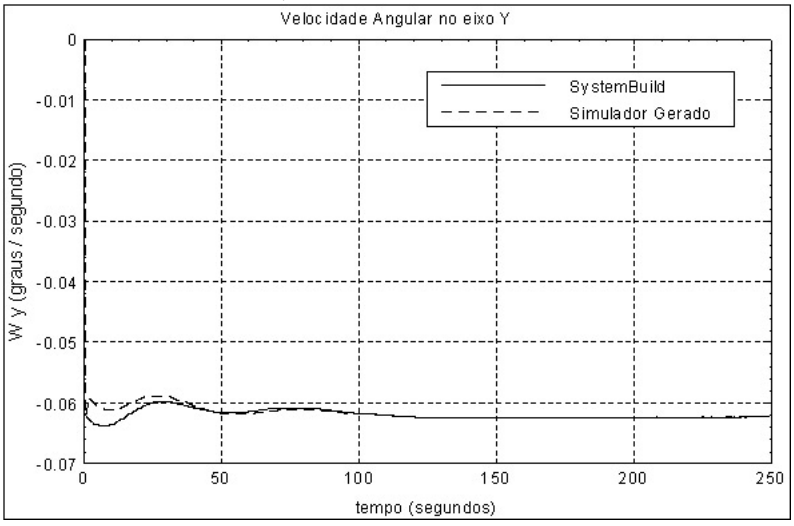
Figura A.24 - Erro entre os valores da velocidade angular do satélite no eixo X (Wx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

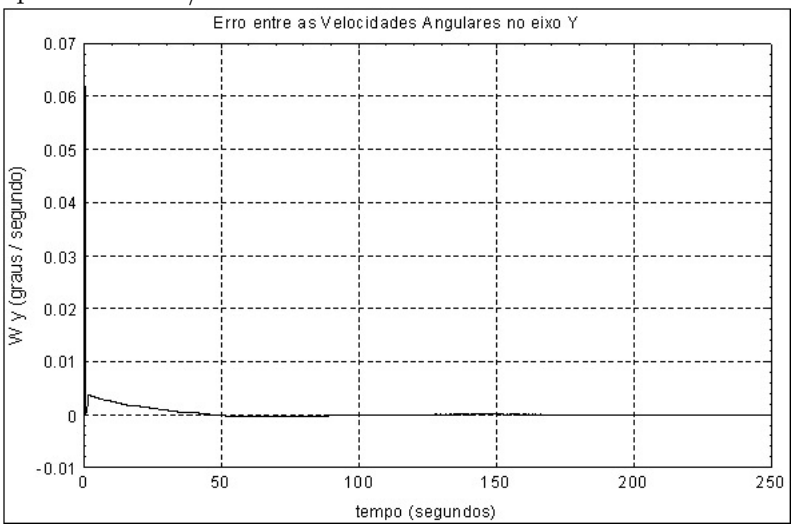
A.5 Velocidade angular no eixo Y (Wy)

Figura A.25 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



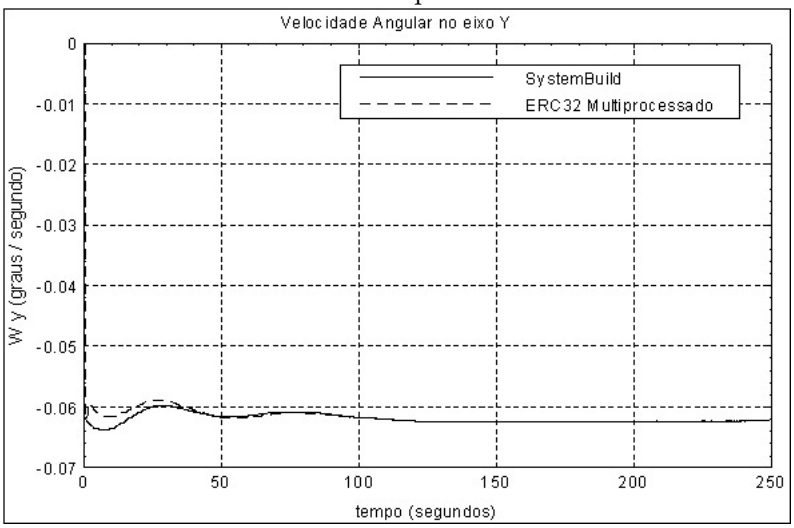
Fonte: Amorim III (2009).

Figura A.26 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



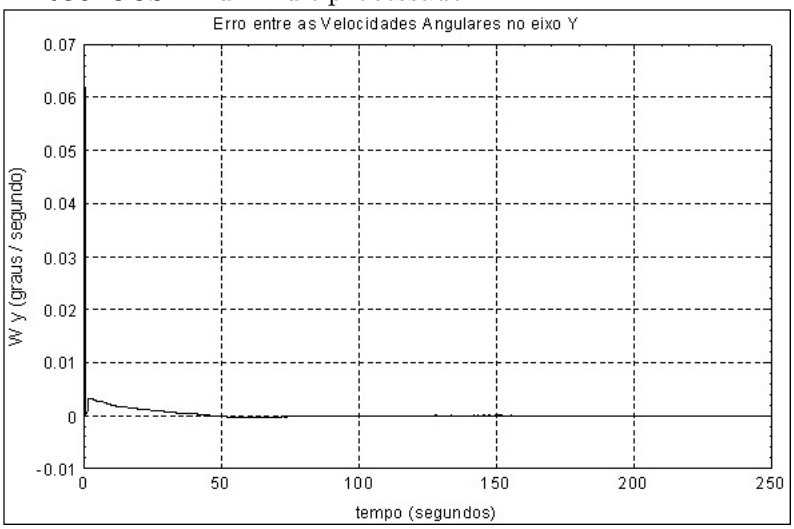
Fonte: Amorim III (2009).

Figura A.27 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



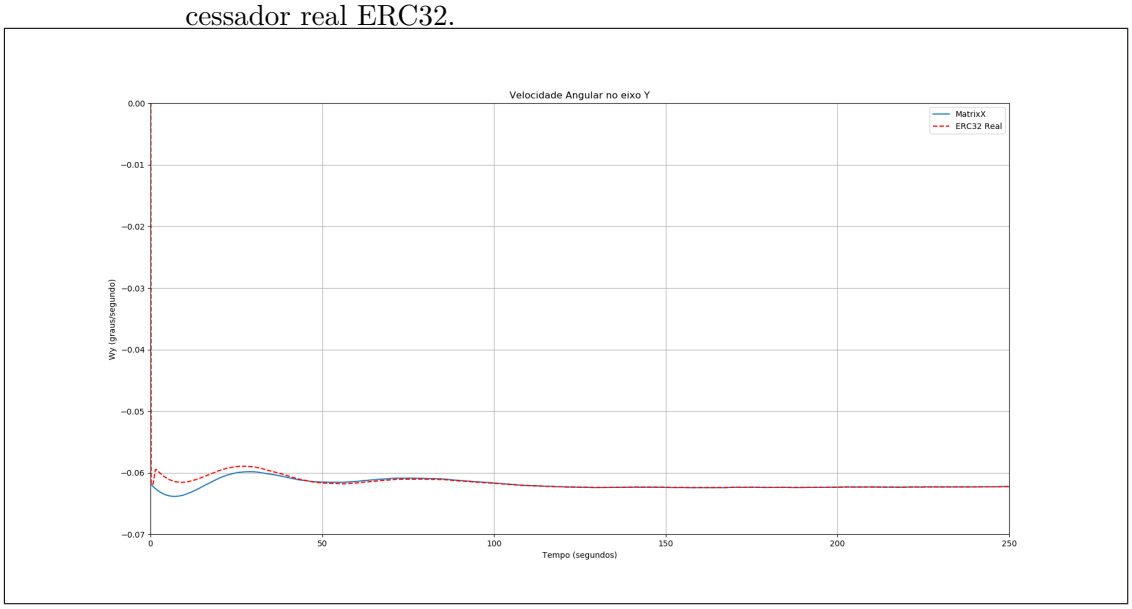
Fonte: Amorim III (2009).

Figura A.28 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



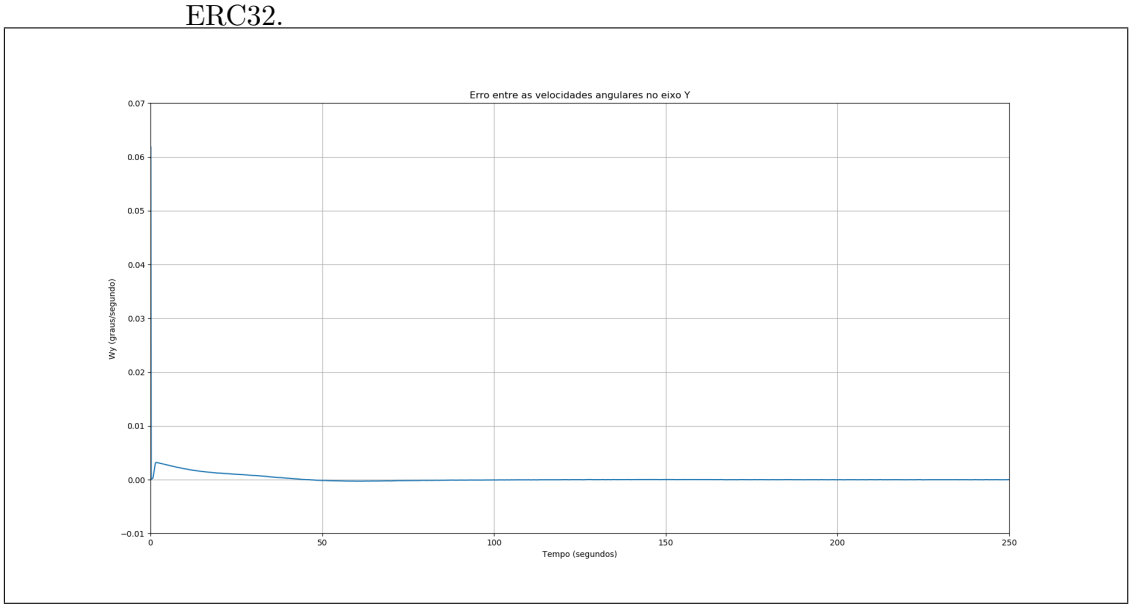
Fonte: Amorim III (2009).

Figura A.29 - Comparação entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

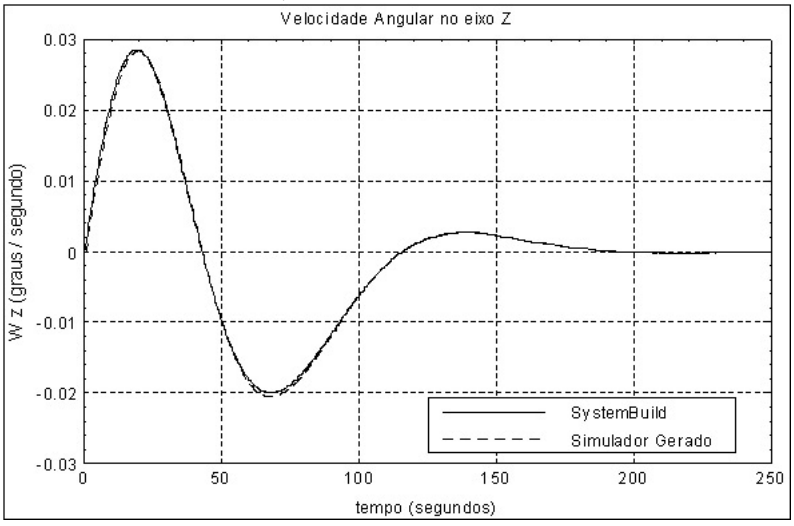
Figura A.30 - Erro entre os valores da velocidade angular do satélite no eixo Y (Wy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

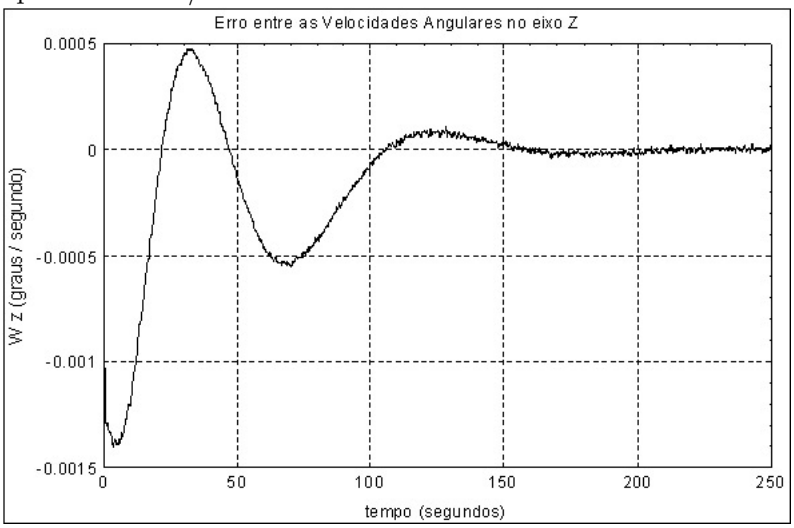
A.6 Velocidade angular no eixo Z (Wz)

Figura A.31 - Comparação entre os valores da velocidade angular do satélite no eixo Z (Wz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



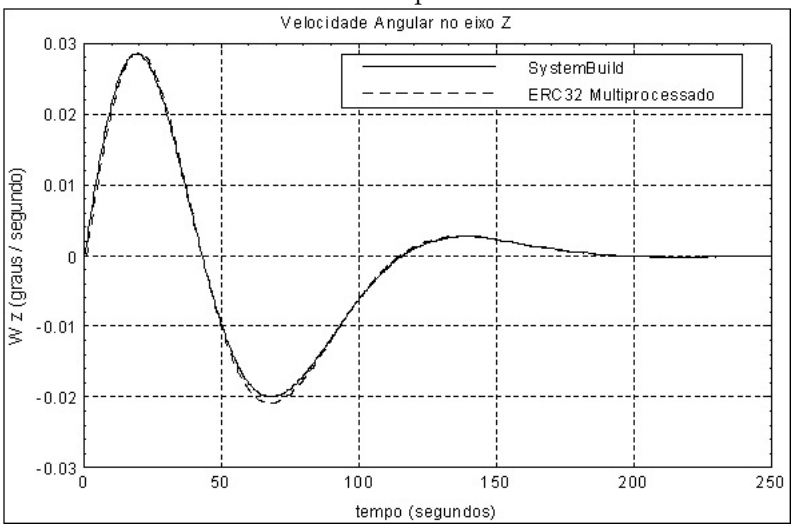
Fonte: Amorim III (2009).

Figura A.32 - Erro entre os valores da velocidade angular do satélite no eixo Z (Wz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



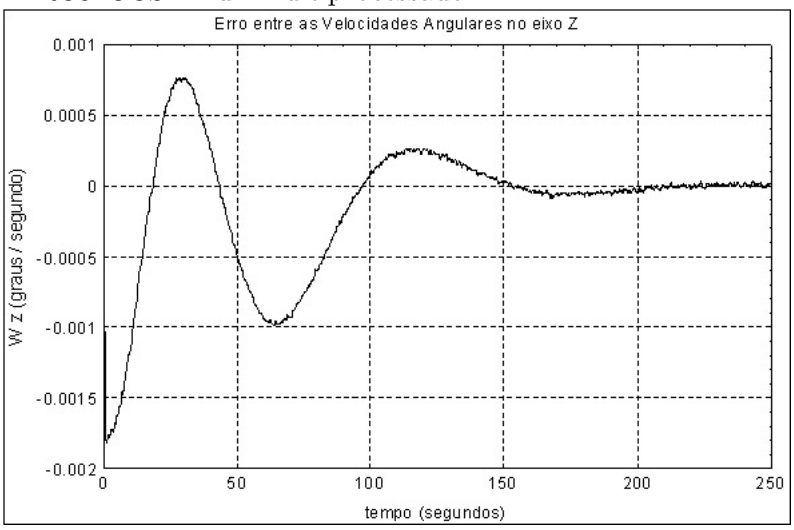
Fonte: Amorim III (2009).

Figura A.33 - Comparação entre os valores da velocidade angular do satélite no eixo Z (Wz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



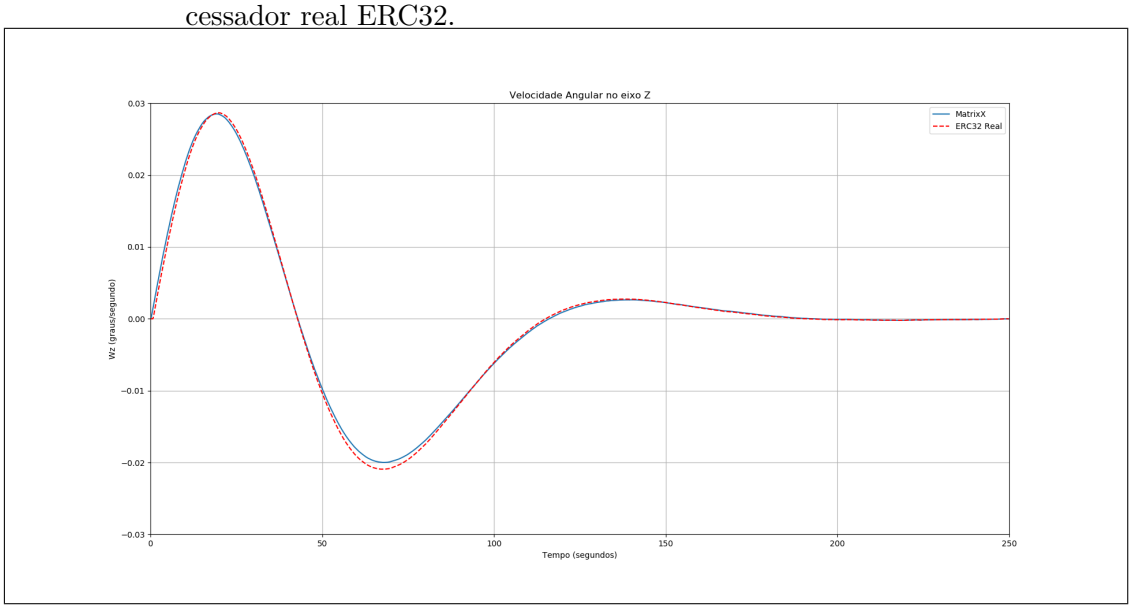
Fonte: Amorim III (2009).

Figura A.34 - Erro entre os valores da velocidade angular do satélite no eixo Z (Wz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



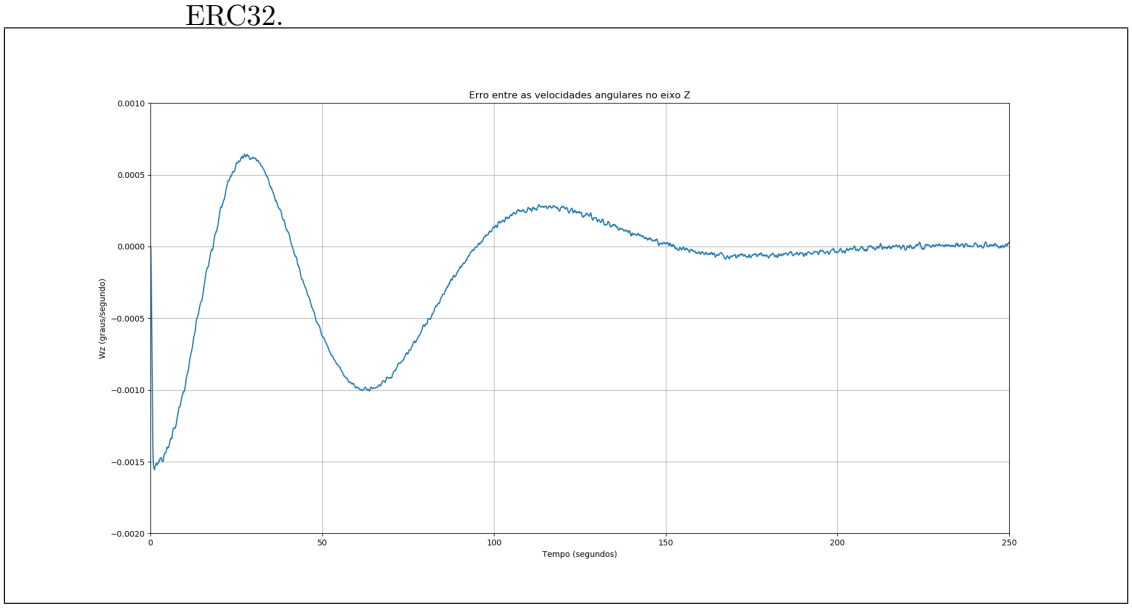
Fonte: Amorim III (2009).

Figura A.35 - Comparação entre os valores da velocidade angular do satélite no eixo Z (Wz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

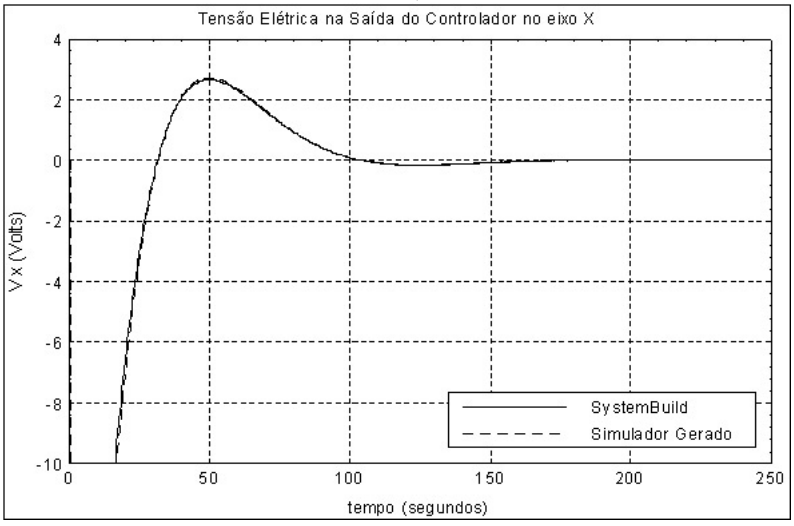
Figura A.36 - Erro entre os valores da velocidade angular do satélite no eixo Z (Wz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

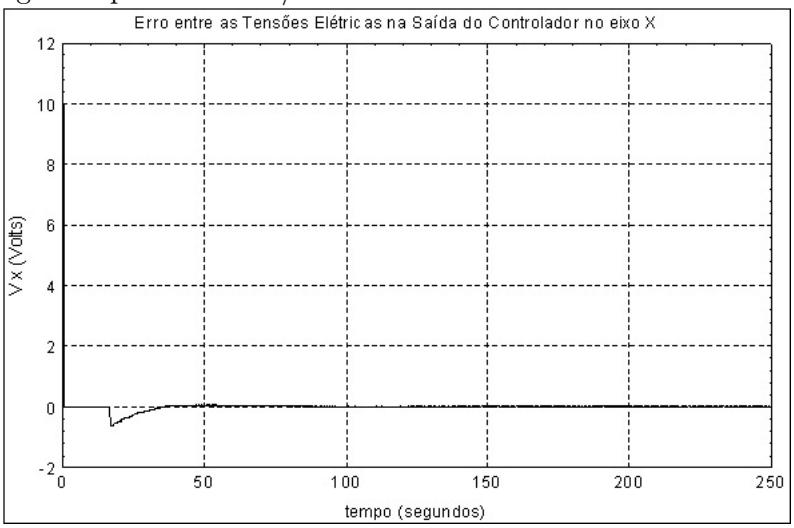
A.7 Saída do Controlador (Vx)

Figura A.37 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



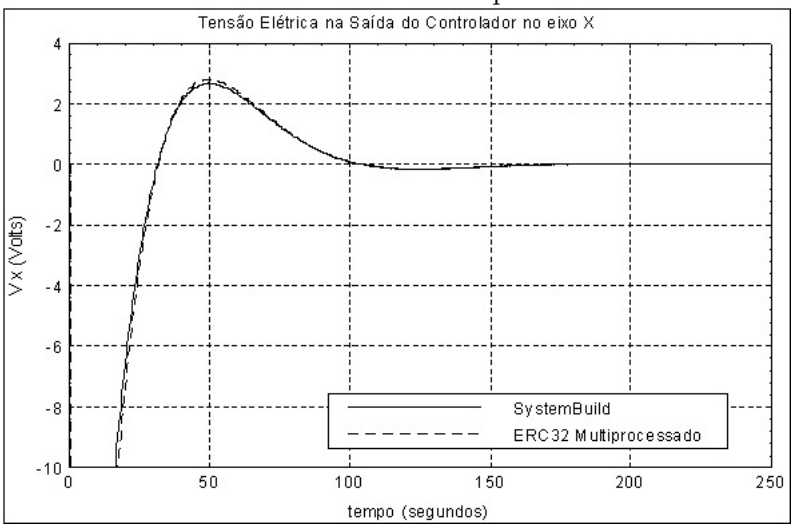
Fonte: Amorim III (2009).

Figura A.38 - Erro entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



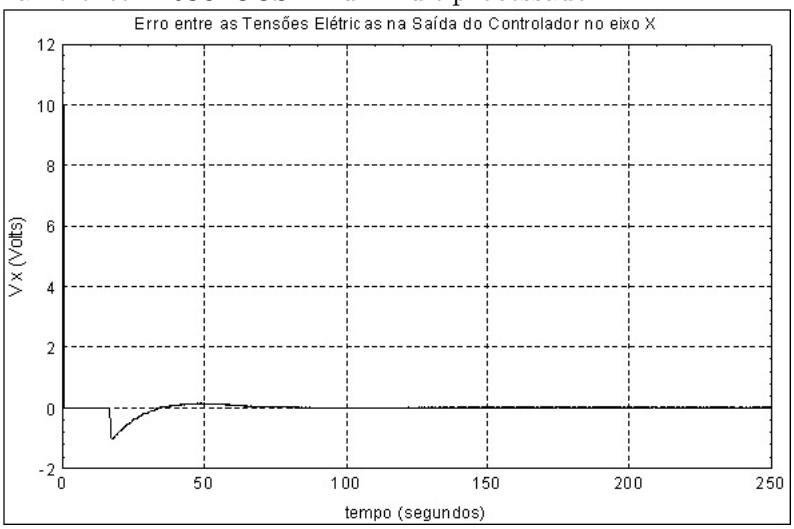
Fonte: Amorim III (2009).

Figura A.39 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



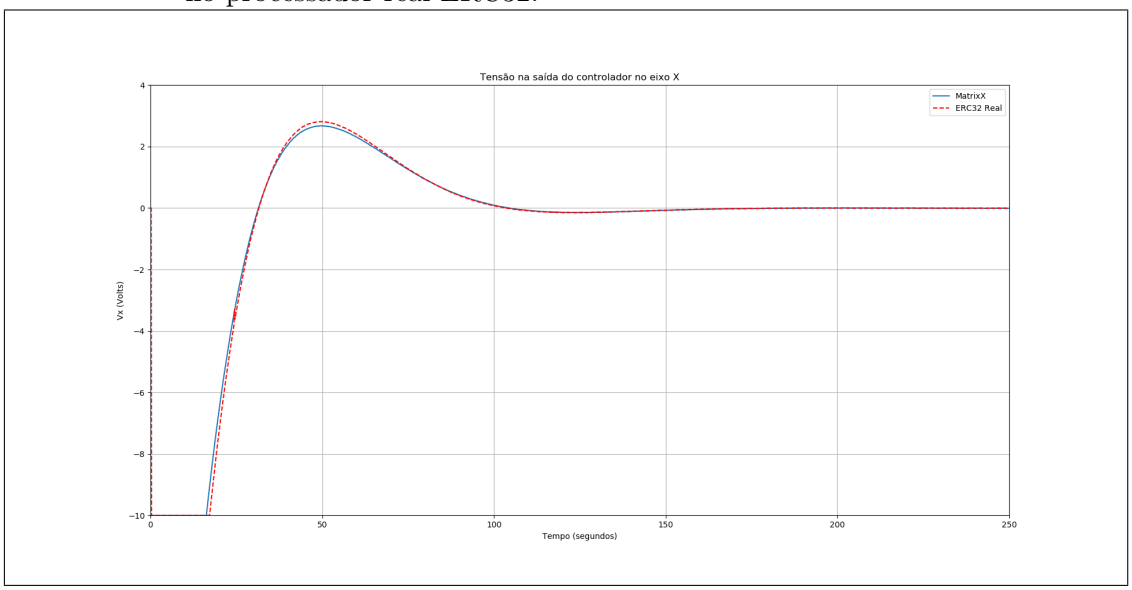
Fonte: Amorim III (2009).

Figura A.40 - Erro entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



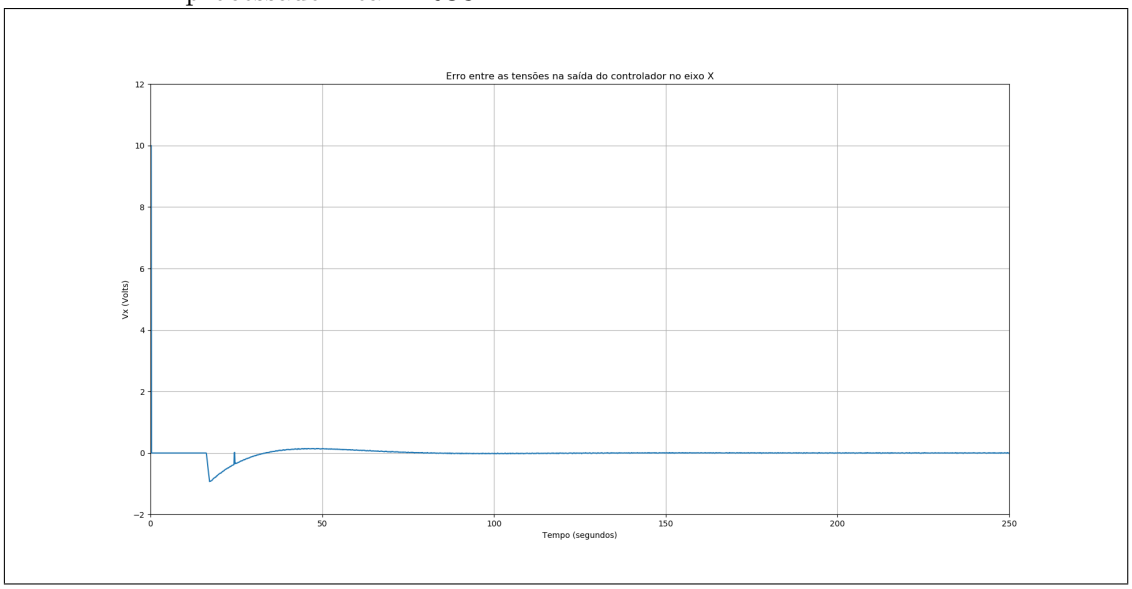
Fonte: Amorim III (2009).

Figura A.41 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

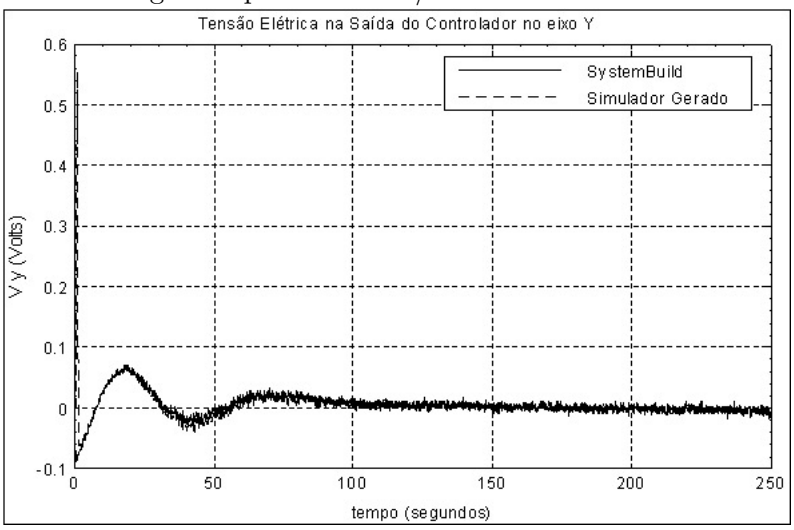
Figura A.42 - Erro entre os valores da tensão elétrica na saída do controlador no eixo X (Vx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

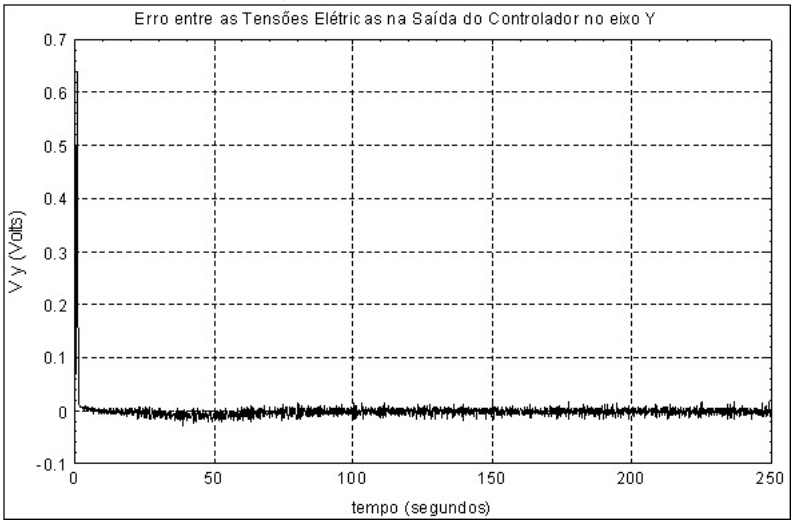
A.8 Saída do Controlador (Vy)

Figura A.43 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



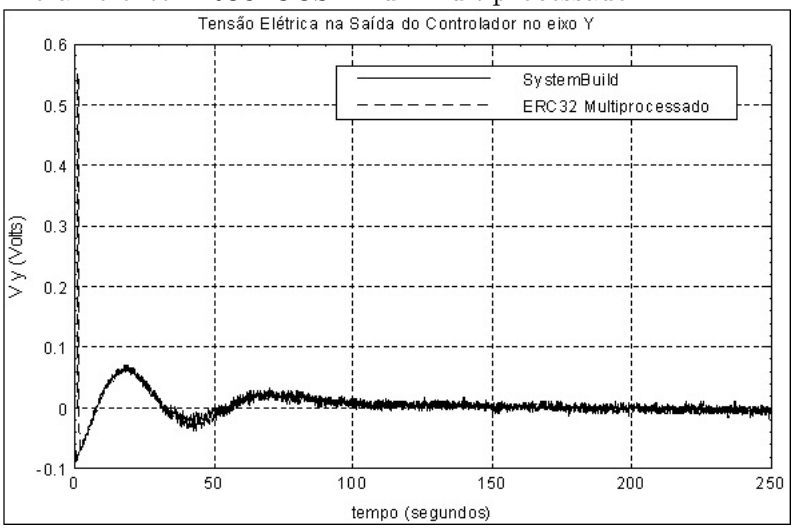
Fonte: Amorim III (2009).

Figura A.44 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



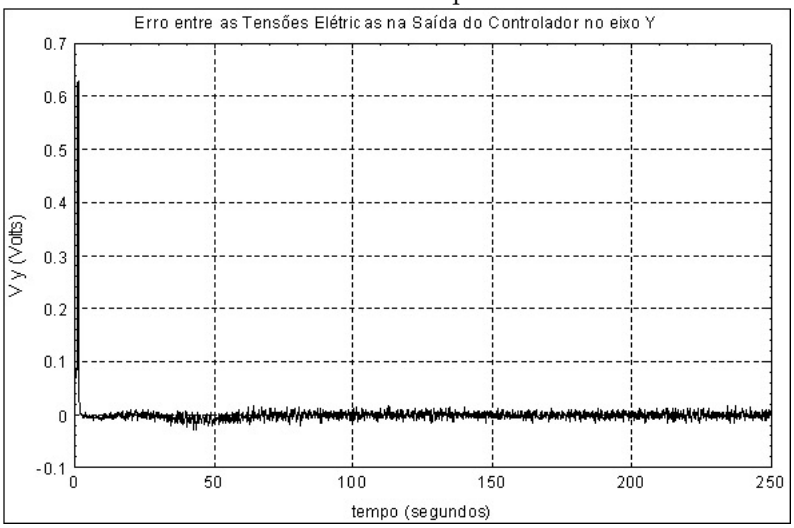
Fonte: Amorim III (2009).

Figura A.45 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



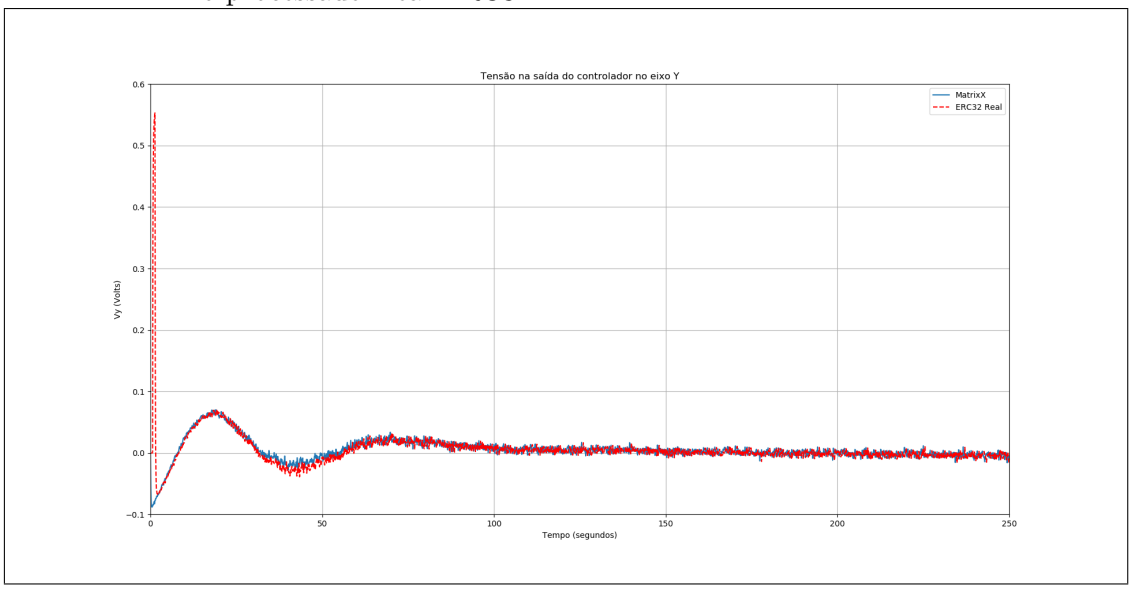
Fonte: Amorim III (2009).

Figura A.46 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



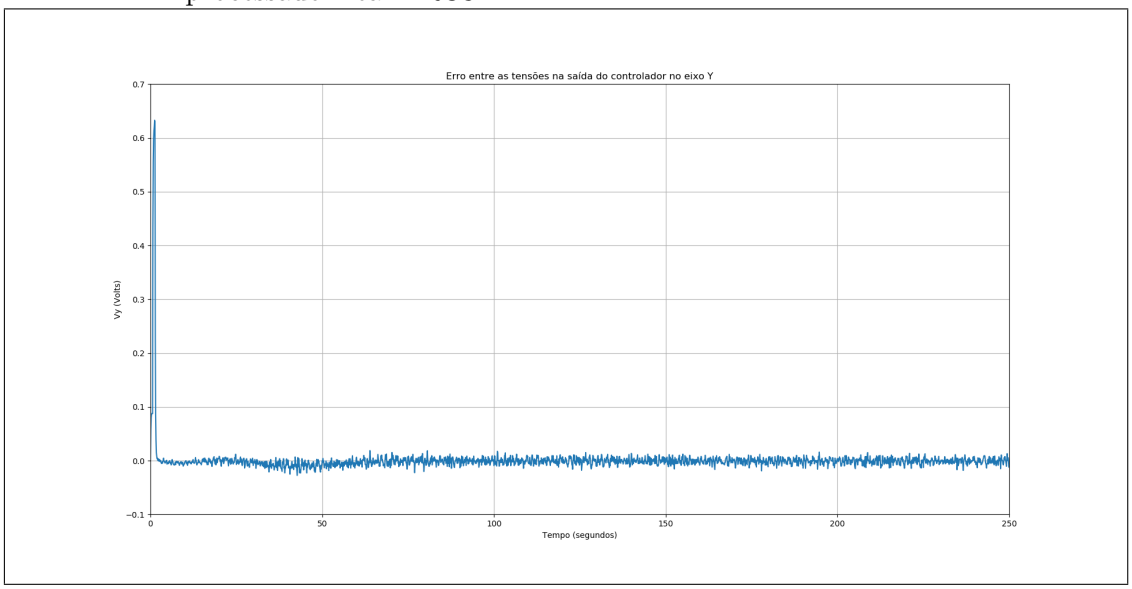
Fonte: Amorim III (2009).

Figura A.47 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

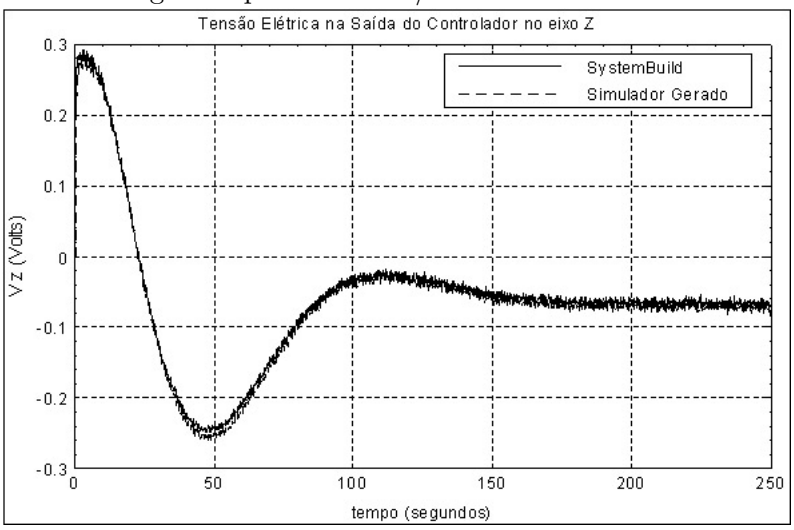
Figura A.48 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Y (Vy) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

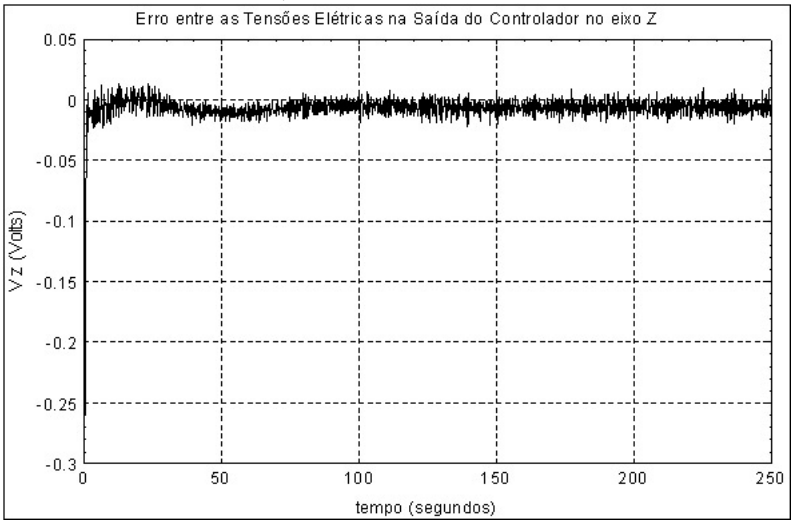
A.9 Saída do Controlador (Vz)

Figura A.49 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



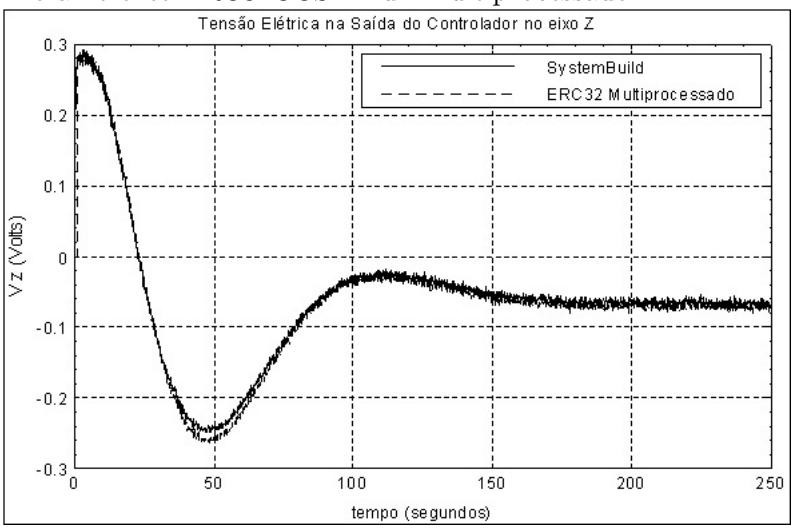
Fonte: Amorim III (2009).

Figura A.50 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



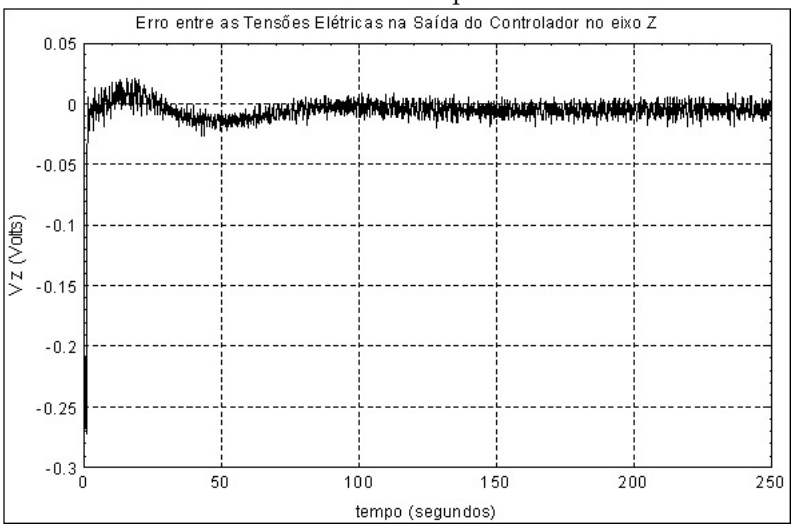
Fonte: Amorim III (2009).

Figura A.51 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



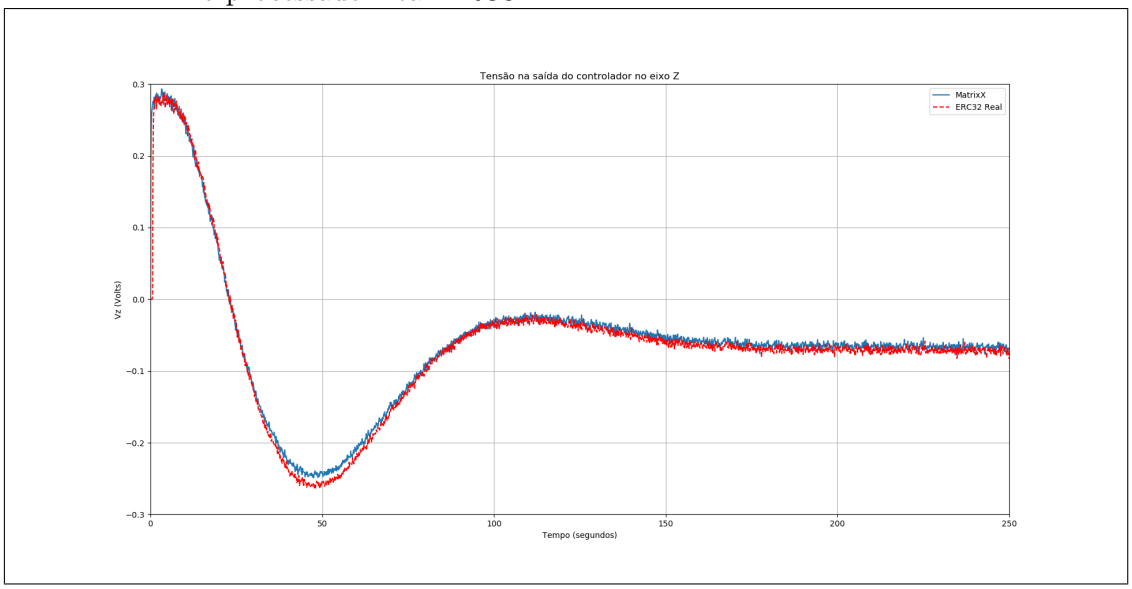
Fonte: Amorim III (2009).

Figura A.52 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



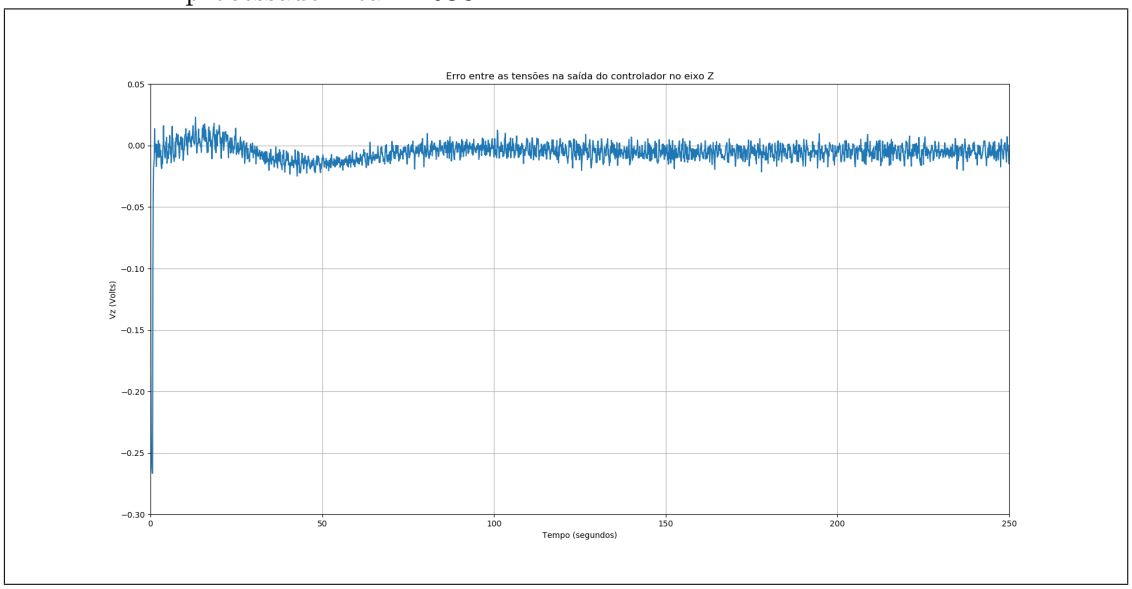
Fonte: Amorim III (2009).

Figura A.53 - Comparação entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

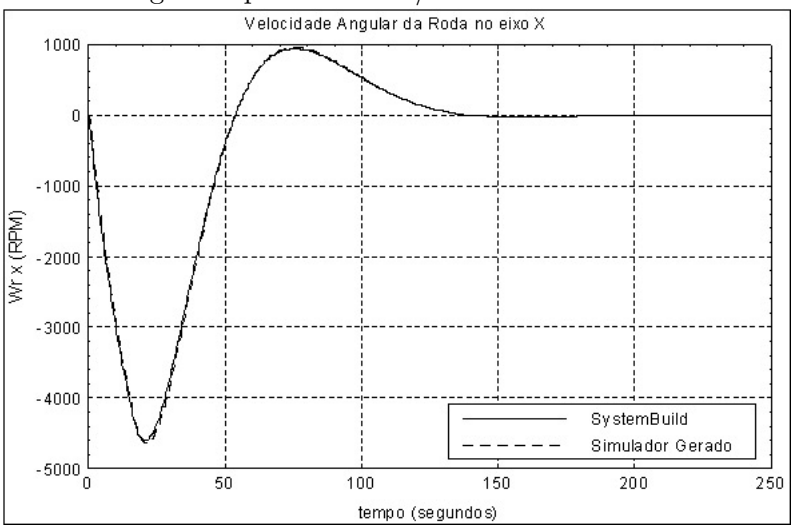
Figura A.54 - Erro entre os valores da tensão elétrica na saída do controlador no eixo Z (Vz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

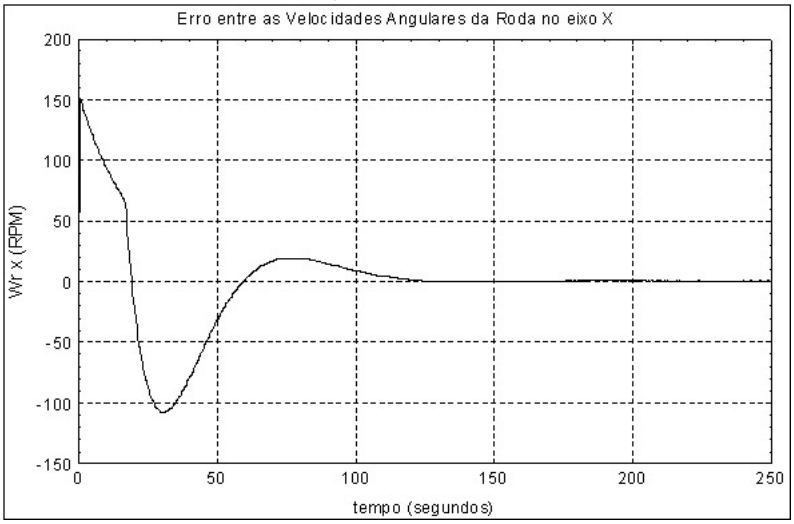
A.10 Velocidade angular da roda de reação no eixo X (Wrx)

Figura A.55 - Comparação entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



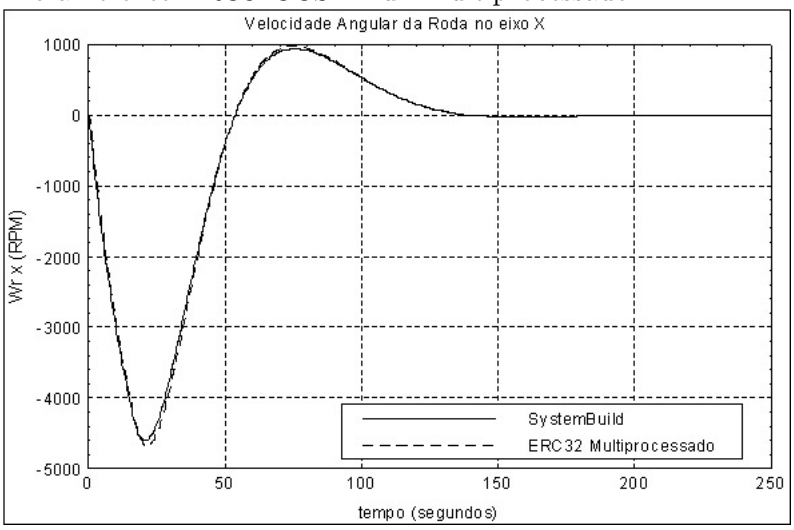
Fonte: Amorim III (2009).

Figura A.56 - Erro entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



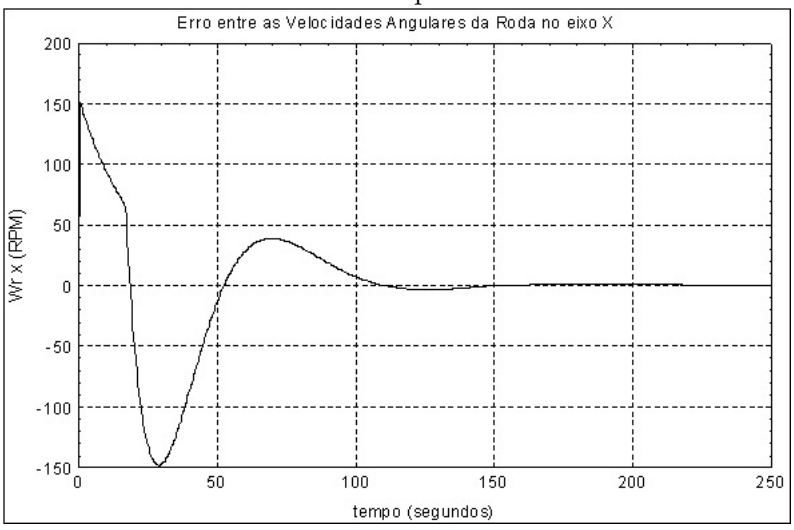
Fonte: Amorim III (2009).

Figura A.57 - Comparação entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



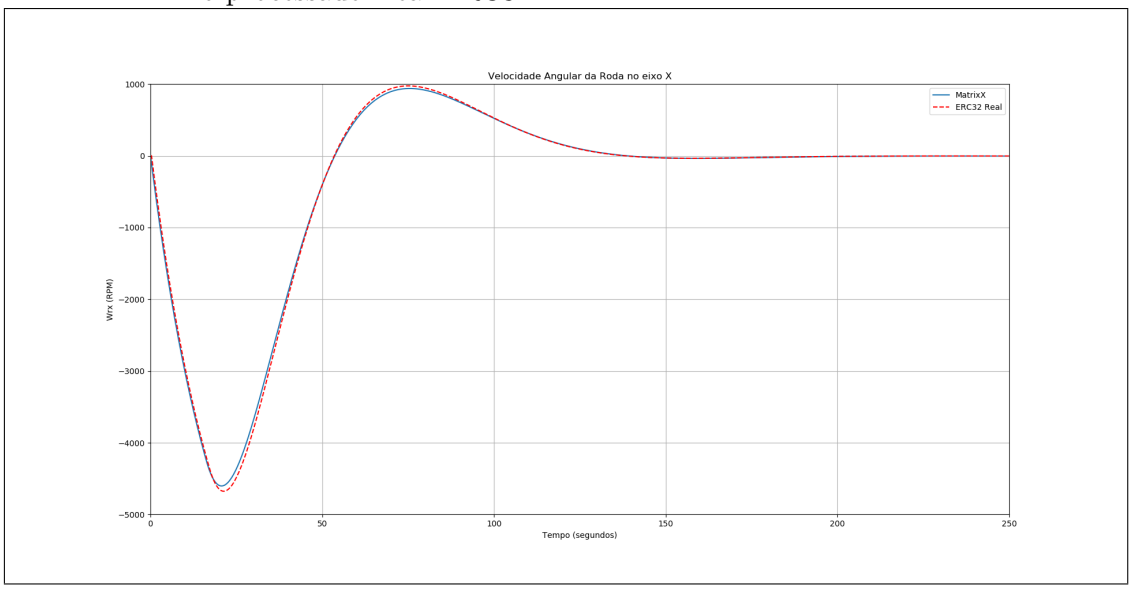
Fonte: Amorim III (2009).

Figura A.58 - Erro entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



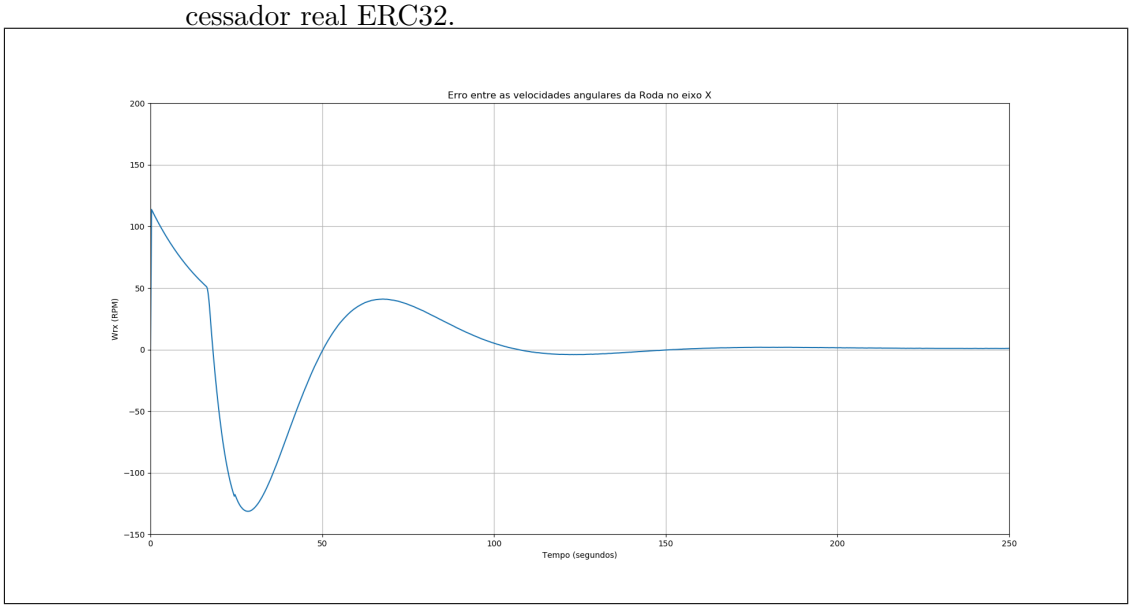
Fonte: Amorim III (2009).

Figura A.59 - Comparação entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

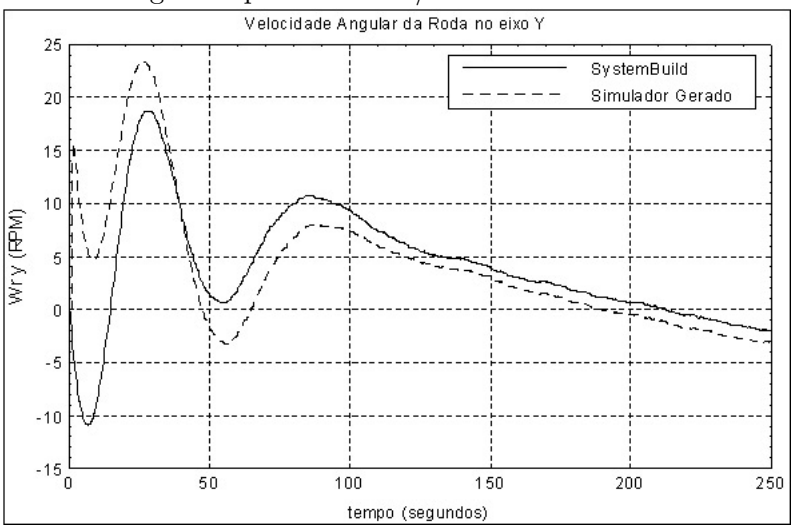
Figura A.60 - Erro entre os valores da velocidade angular da roda de reação no eixo X (Wrx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

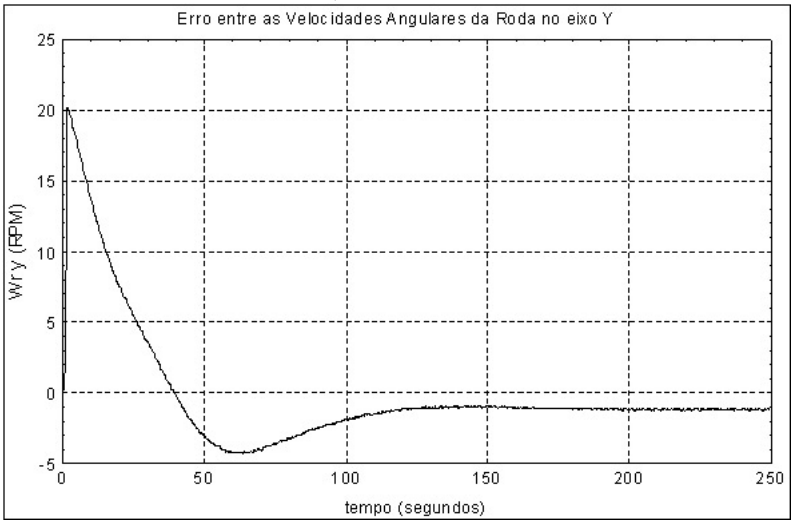
A.11 Velocidade angular da roda de reação no eixo Y (Wry)

Figura A.61 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



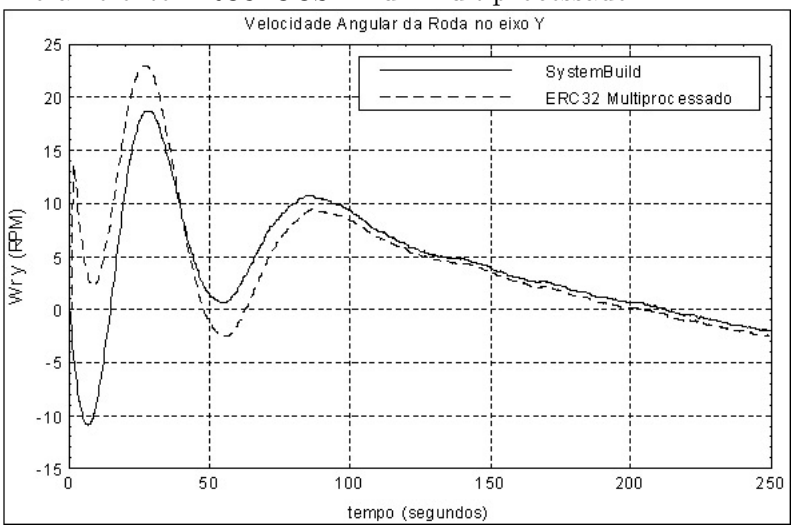
Fonte: Amorim III (2009).

Figura A.62 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



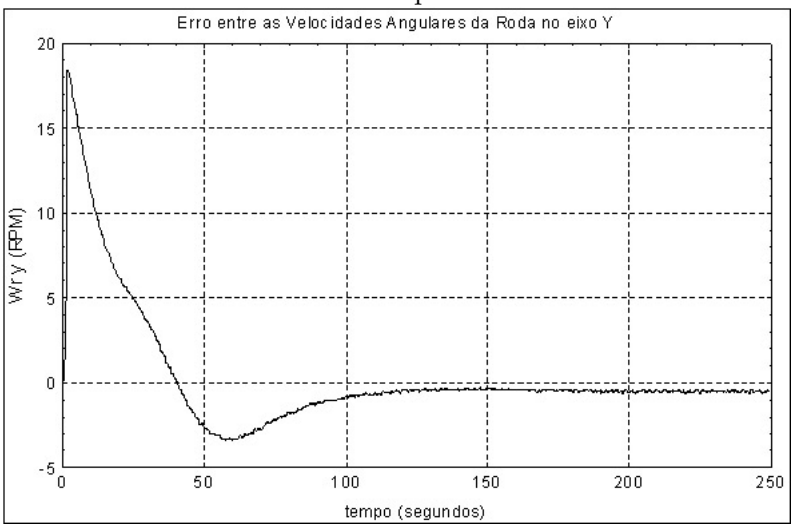
Fonte: Amorim III (2009).

Figura A.63 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



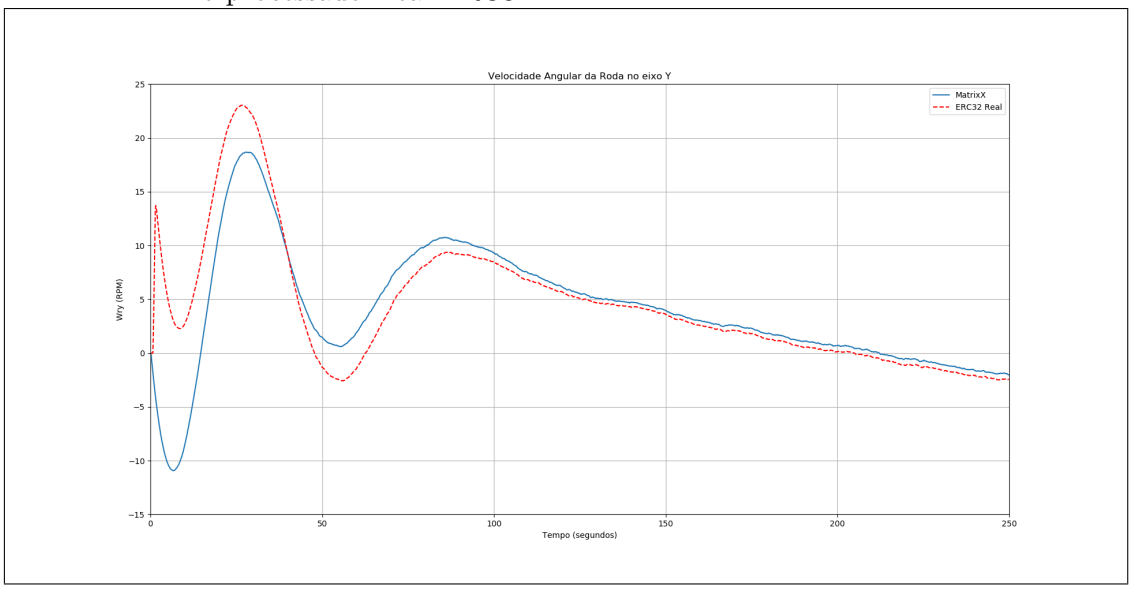
Fonte: Amorim III (2009).

Figura A.64 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



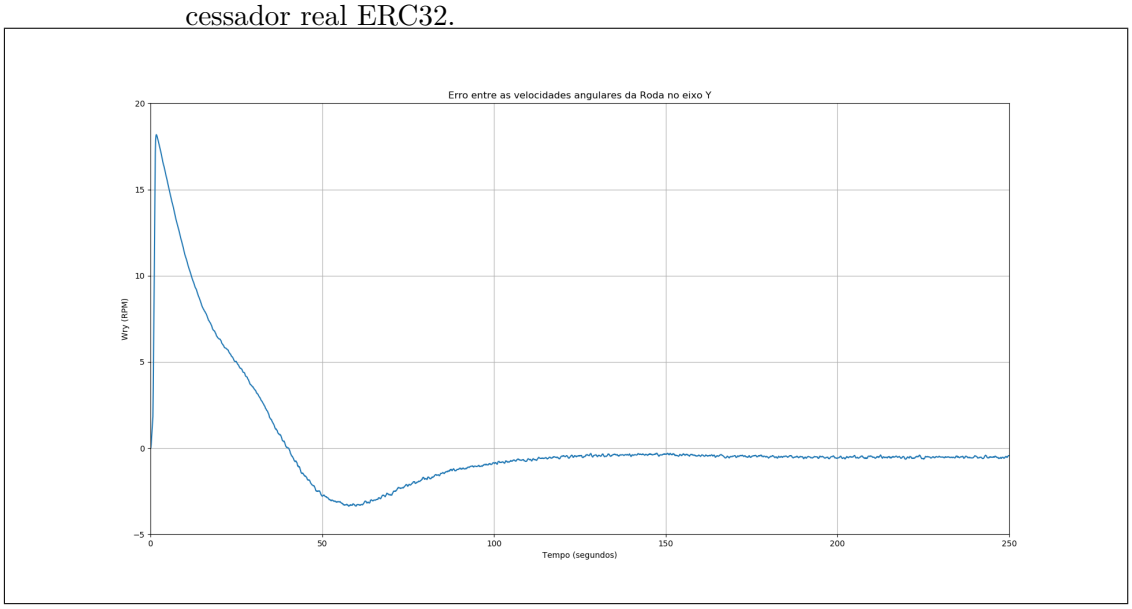
Fonte: Amorim III (2009).

Figura A.65 - Comparação entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

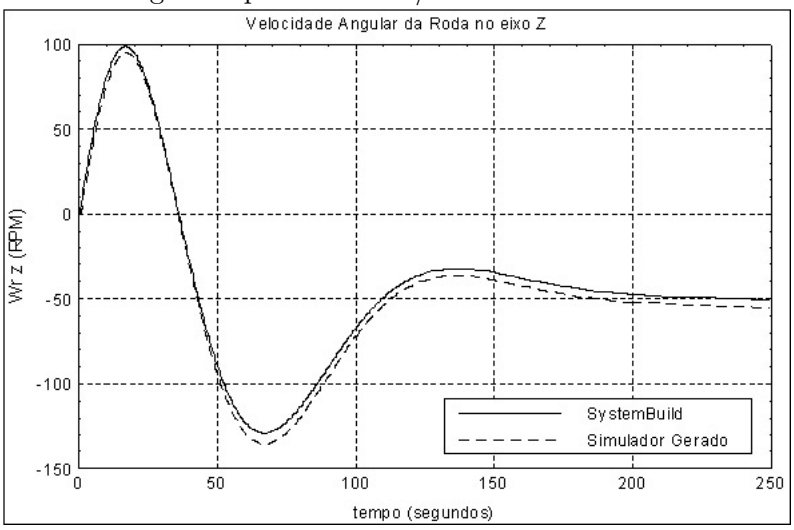
Figura A.66 - Erro entre os valores da velocidade angular da roda de reação no eixo Y (Wry) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

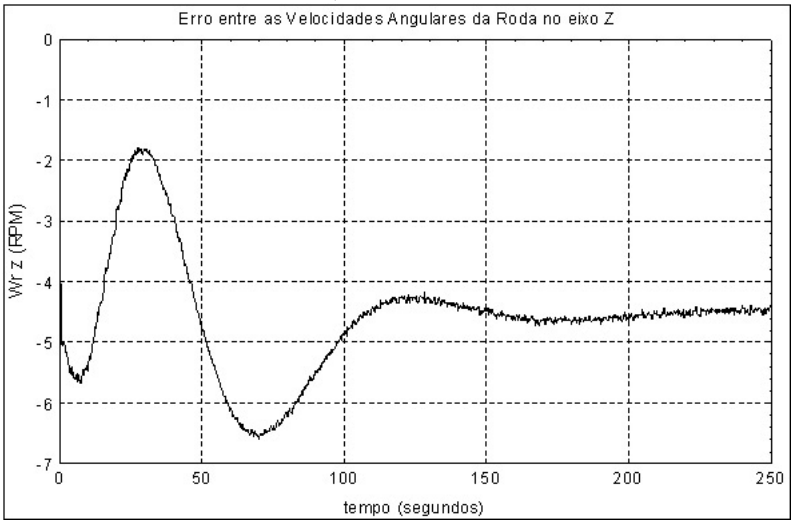
A.12 Velocidade angular da roda de reação no eixo Z (Wrz)

Figura A.67 - Comparação entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



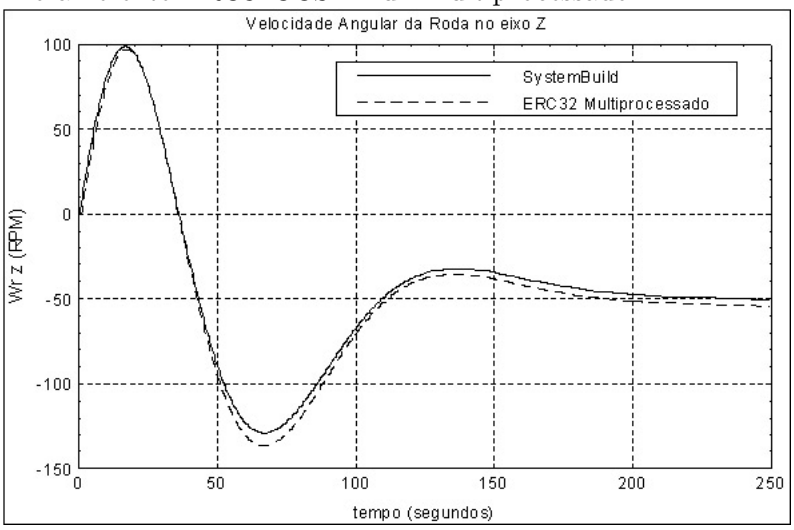
Fonte: Amorim III (2009).

Figura A.68 - Erro entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



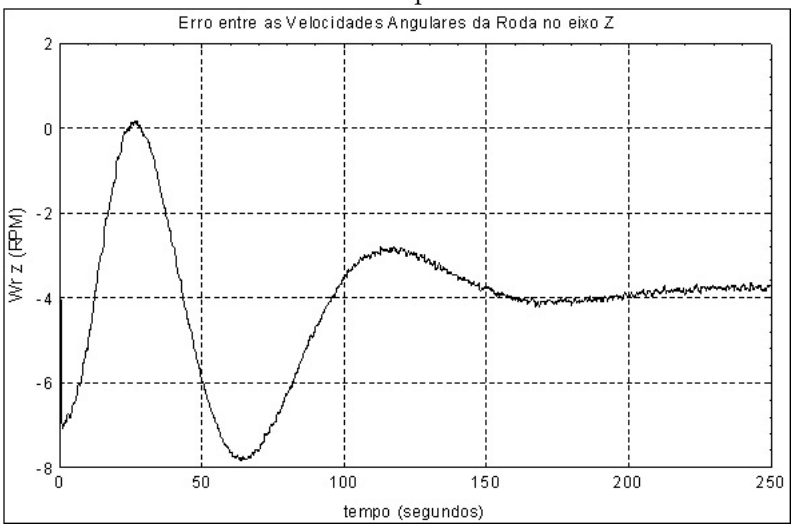
Fonte: Amorim III (2009).

Figura A.69 - Comparação entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



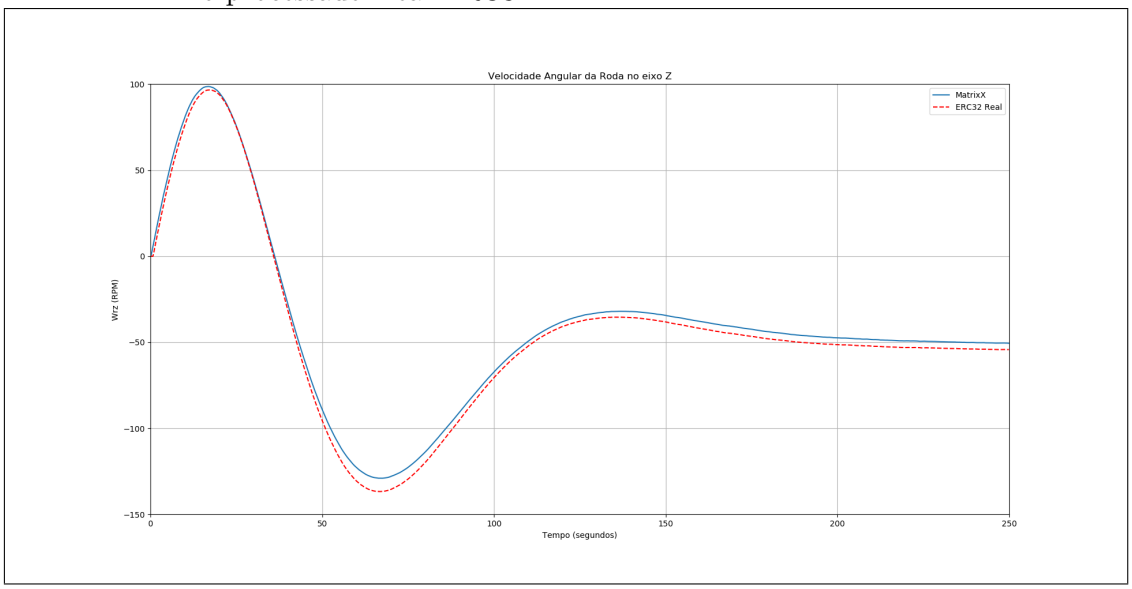
Fonte: Amorim III (2009).

Figura A.70 - Erro entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



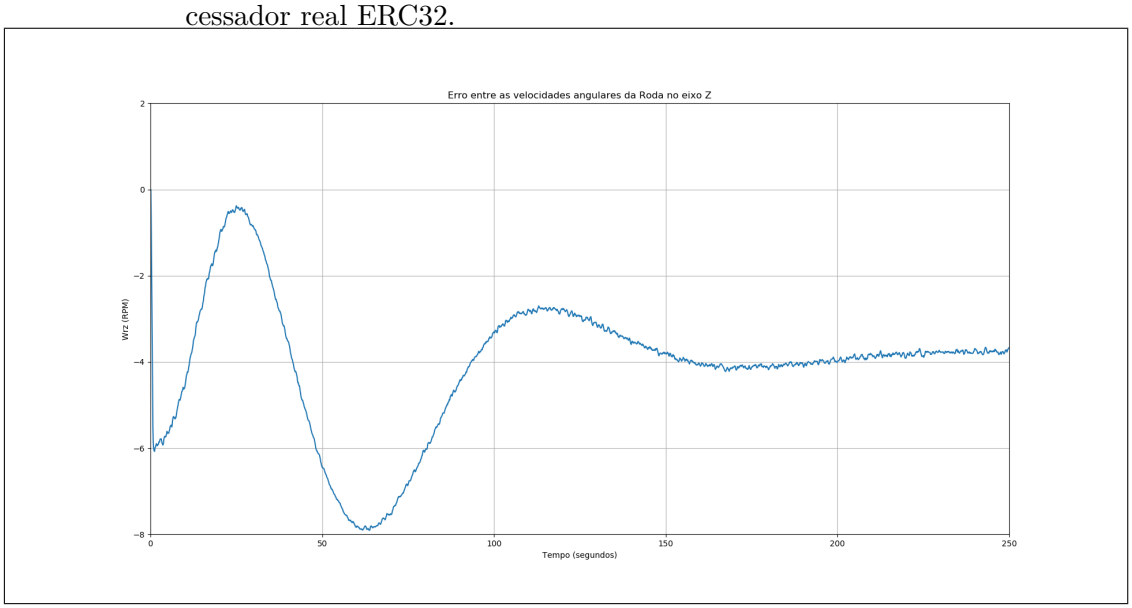
Fonte: Amorim III (2009).

Figura A.71 - Comparação entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

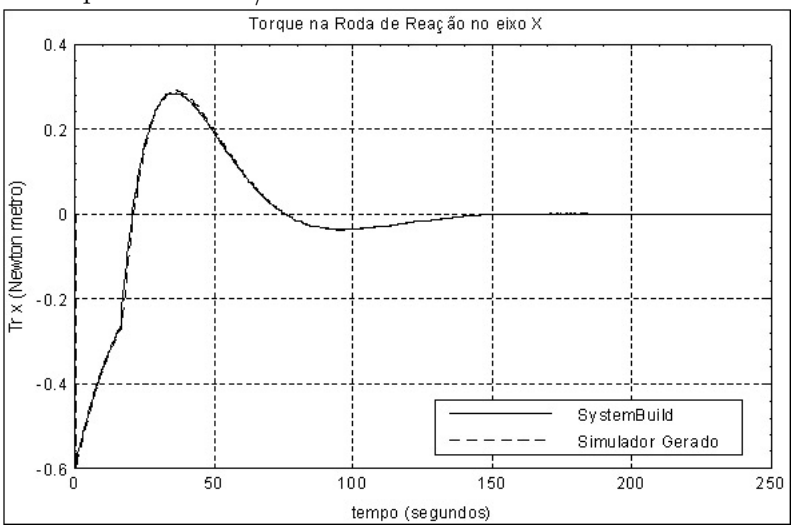
Figura A.72 - Erro entre os valores da velocidade angular da roda de reação no eixo Z (Wrz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

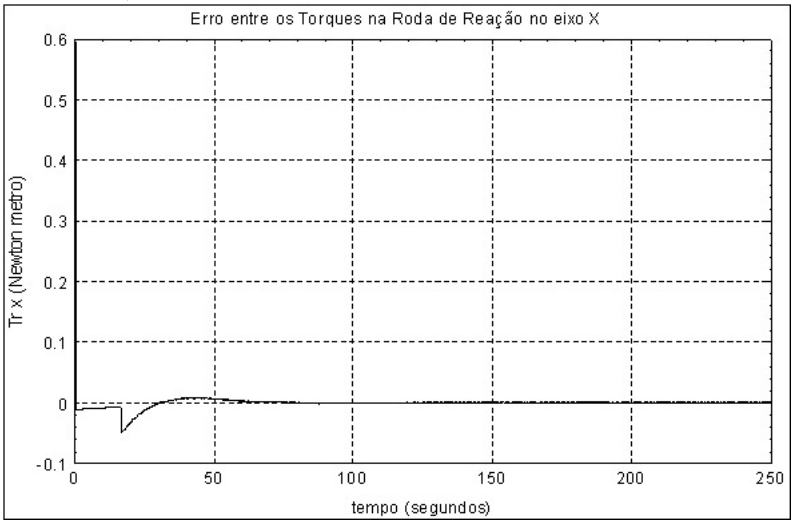
A.13 Torque da roda de reação no eixo X (Trx)

Figura A.73 - Comparação entre os valores do torque da roda de reação no eixo X (Trx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



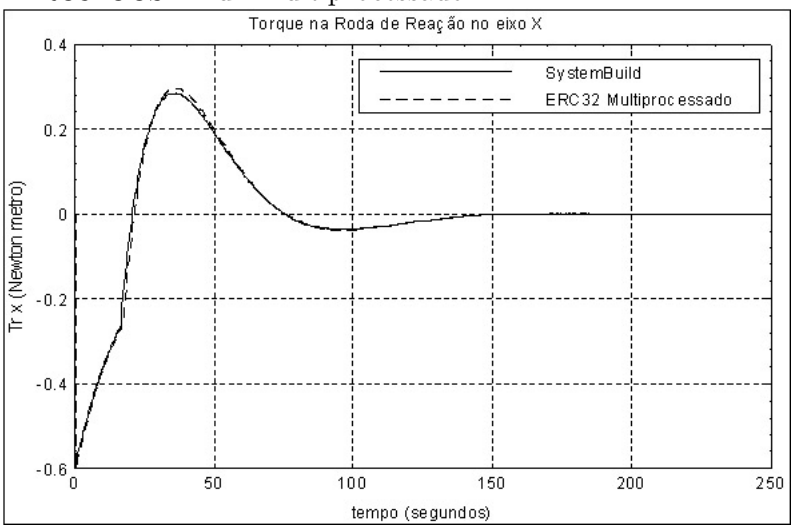
Fonte: Amorim III (2009).

Figura A.74 - Erro entre os valores do torque da roda de reação no eixo X (Trx) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



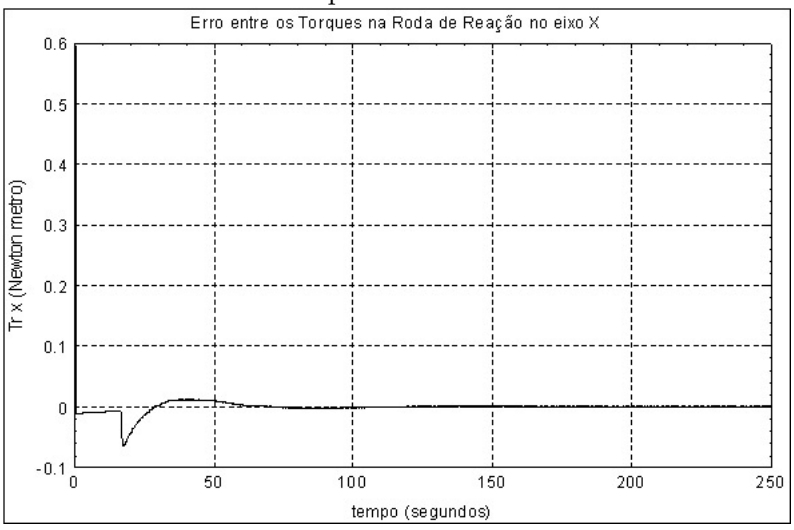
Fonte: Amorim III (2009).

Figura A.75 - Comparação entre os valores do torque da roda de reação no eixo X (Trx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



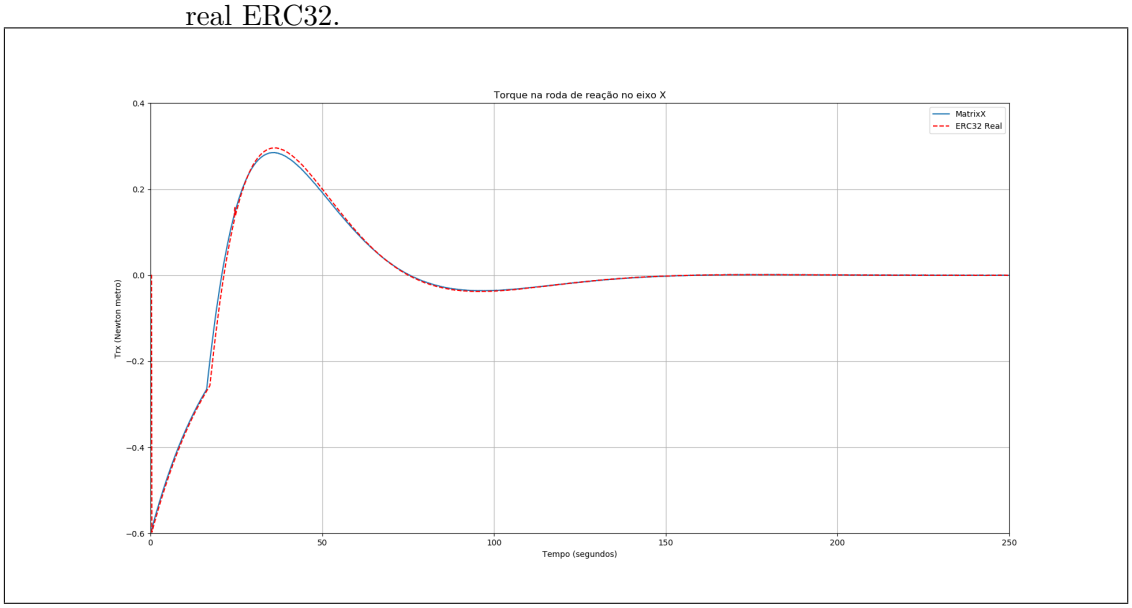
Fonte: Amorim III (2009).

Figura A.76 - Erro entre os valores do torque da roda de reação no eixo X (Trx) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



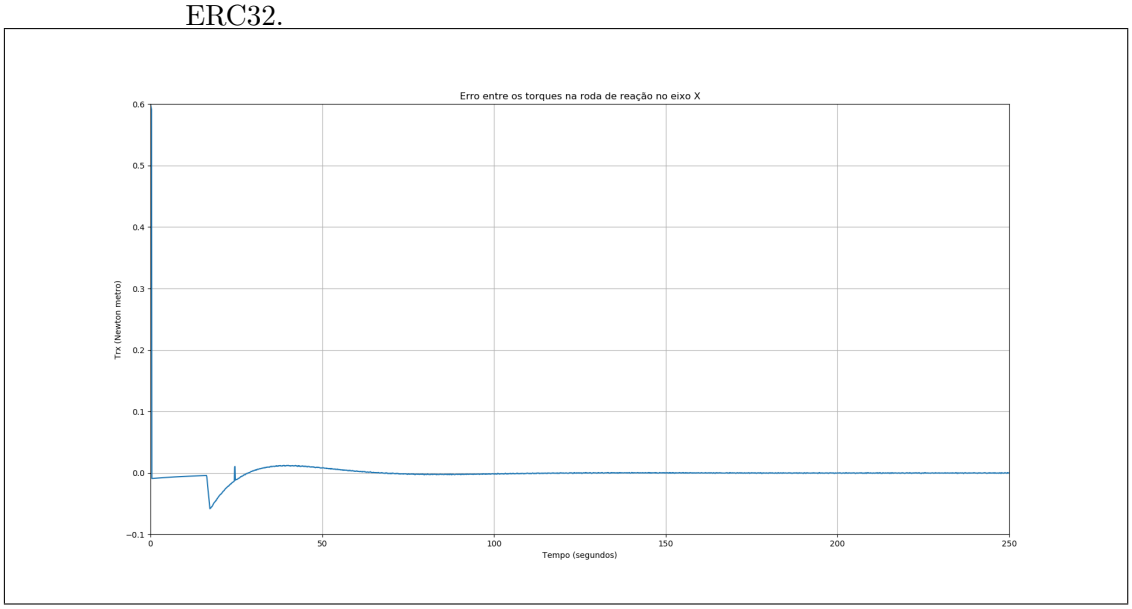
Fonte: Amorim III (2009).

Figura A.77 - Comparação entre os valores do torque da roda de reação no eixo X (Trx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

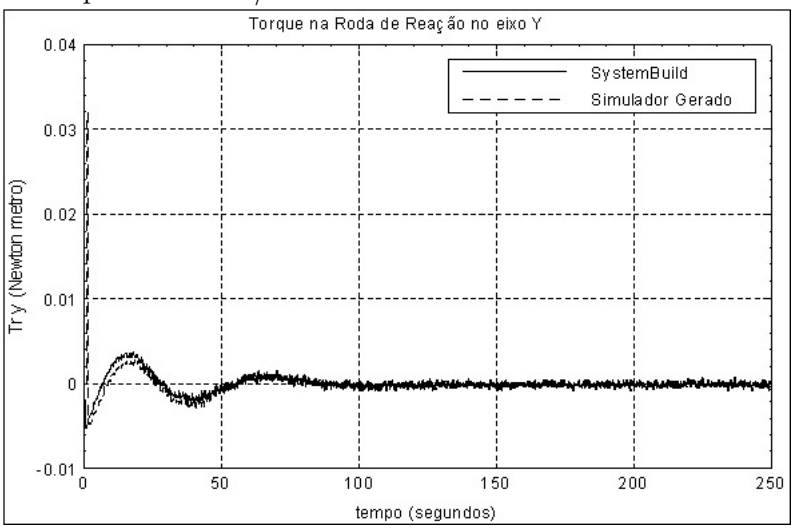
Figura A.78 - Erro entre os valores do torque da roda de reação no eixo X (Trx) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

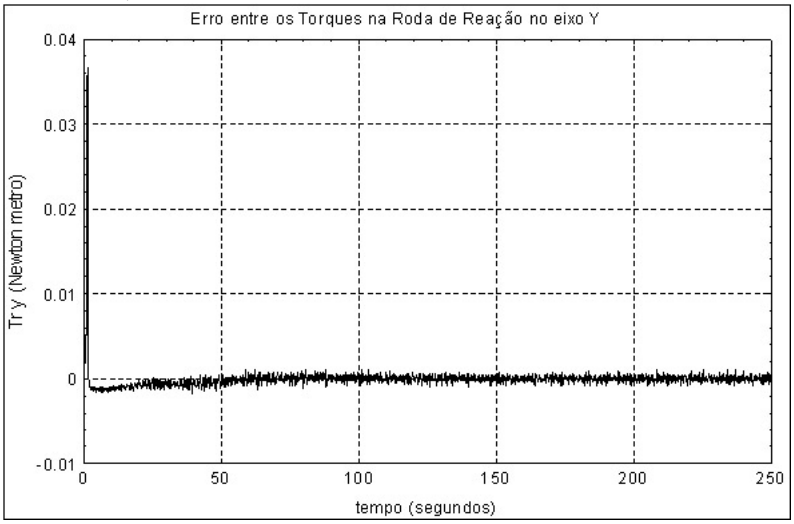
A.14 Torque da roda de reação no eixo Y (Try)

Figura A.79 - Comparação entre os valores do torque da roda de reação no eixo Y (Try) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



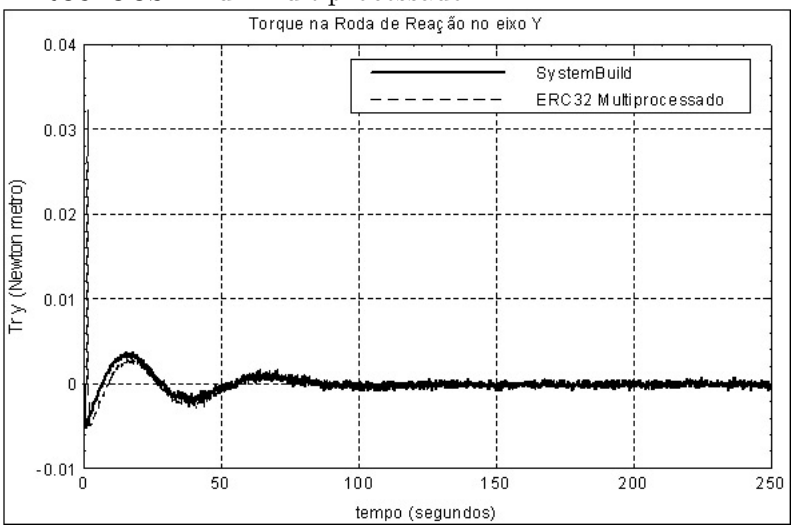
Fonte: Amorim III (2009).

Figura A.80 - Erro entre os valores do torque da roda de reação no eixo Y (Try) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



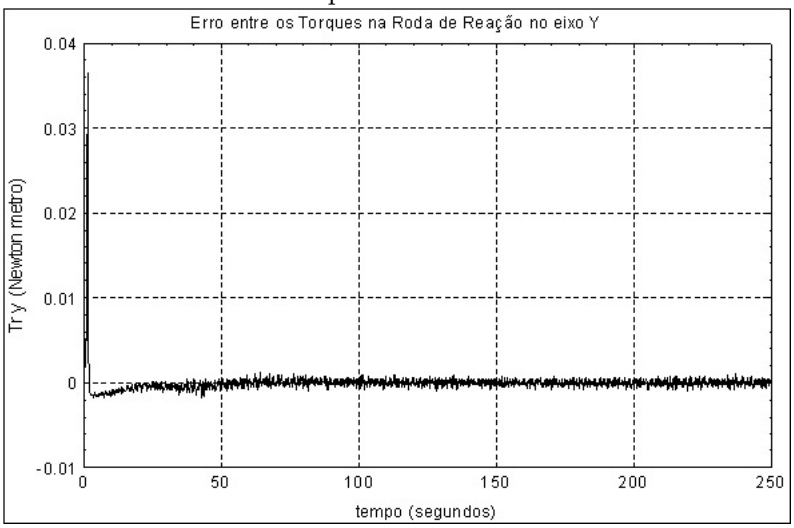
Fonte: Amorim III (2009).

Figura A.81 - Comparação entre os valores do torque da roda de reação no eixo Y (Try) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



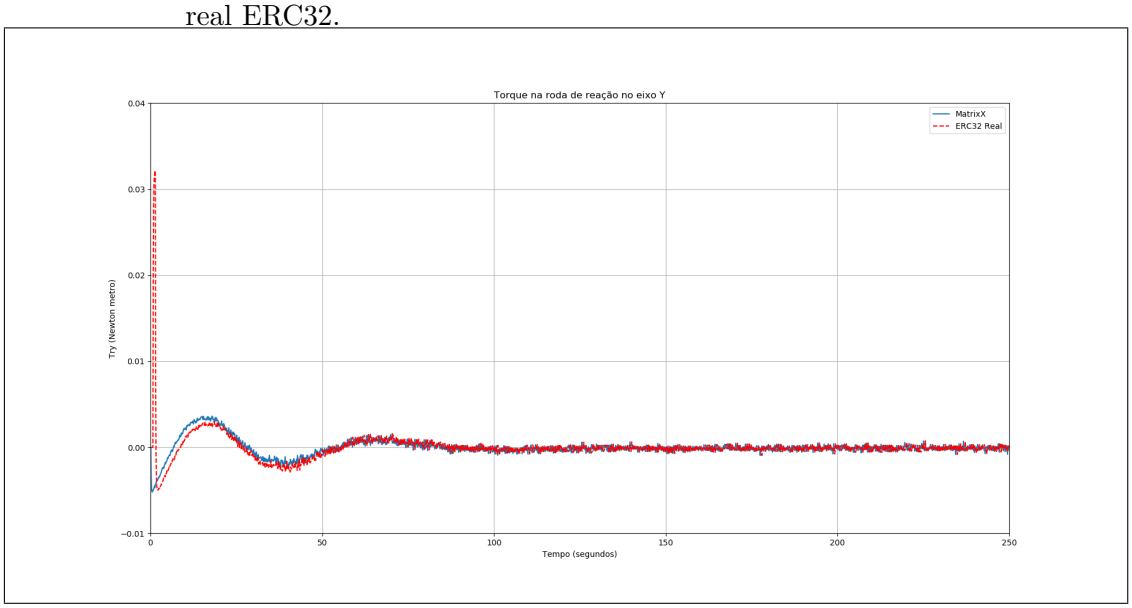
Fonte: Amorim III (2009).

Figura A.82 - Erro entre os valores do torque da roda de reação no eixo Y (Try) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



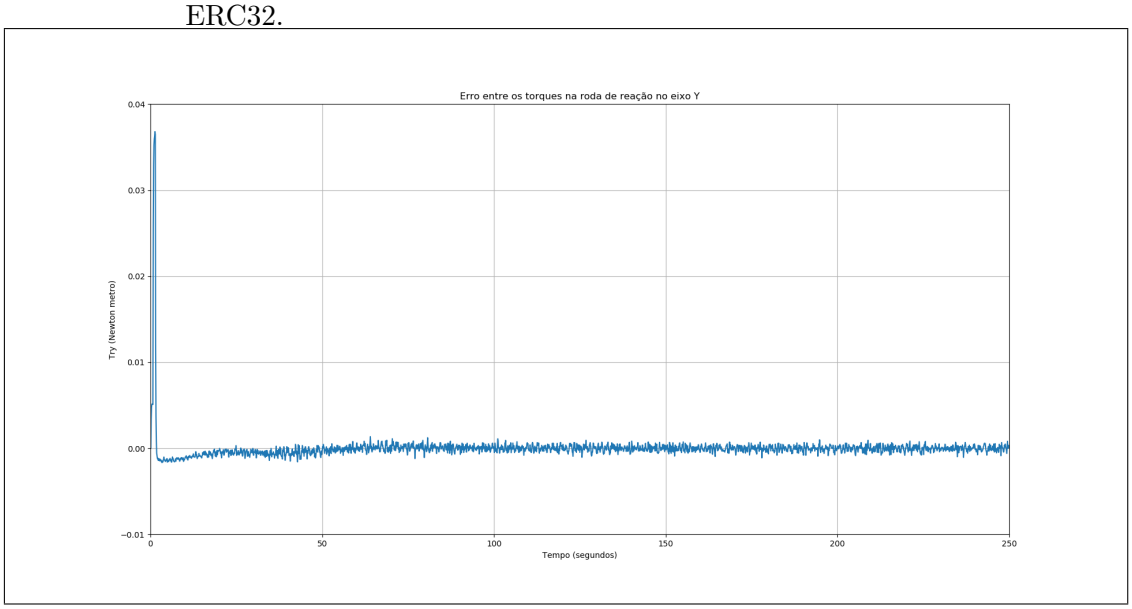
Fonte: Amorim III (2009).

Figura A.83 - Comparação entre os valores do torque da roda de reação no eixo Y (Try) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

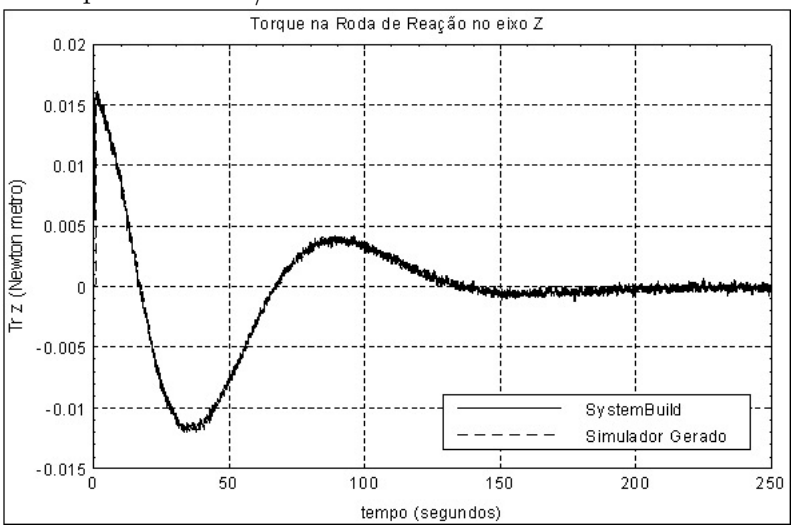
Figura A.84 - Erro entre os valores do torque da roda de reação no eixo Y (Try) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

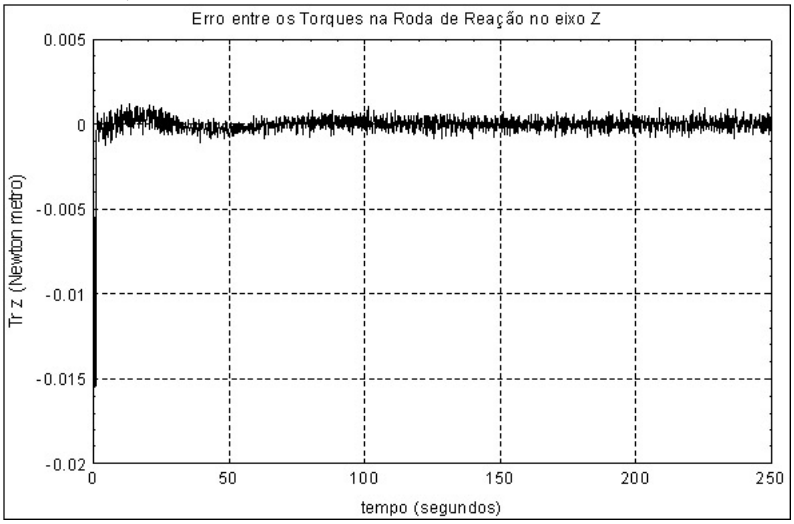
A.15 Torque da roda de reação no eixo Z (Trz)

Figura A.85 - Comparação entre os valores do torque da roda de reação no eixo Z (Trz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



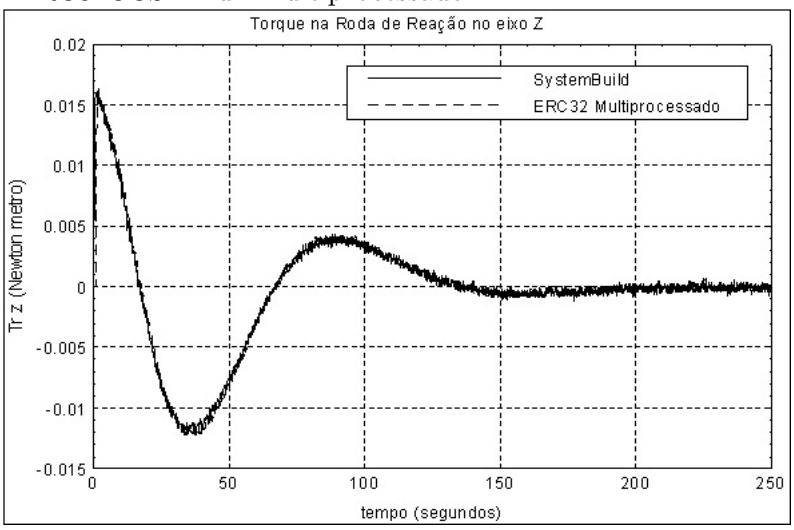
Fonte: Amorim III (2009).

Figura A.86 - Erro entre os valores do torque da roda de reação no eixo Z (Trz) para as simulações: no ambiente MatrixX/SystemBuild e no simulador gerado pelo MatrixX/AutoCode.



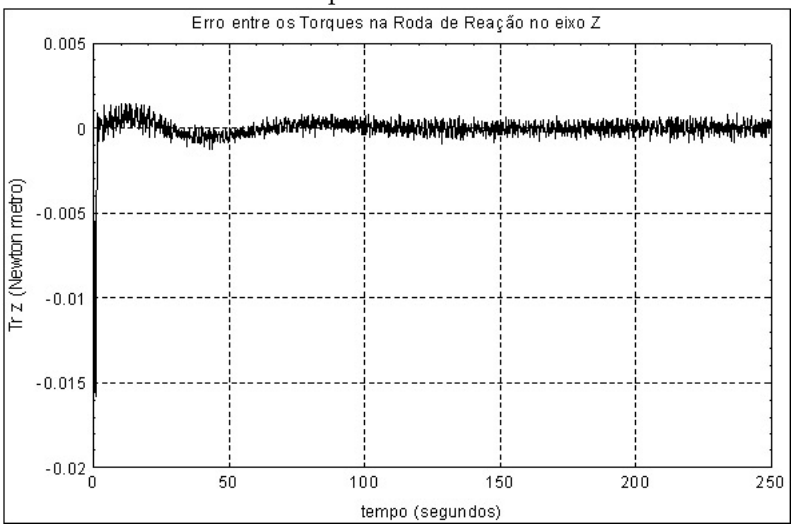
Fonte: Amorim III (2009).

Figura A.87 - Comparação entre os valores do torque da roda de reação no eixo Z (Trz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



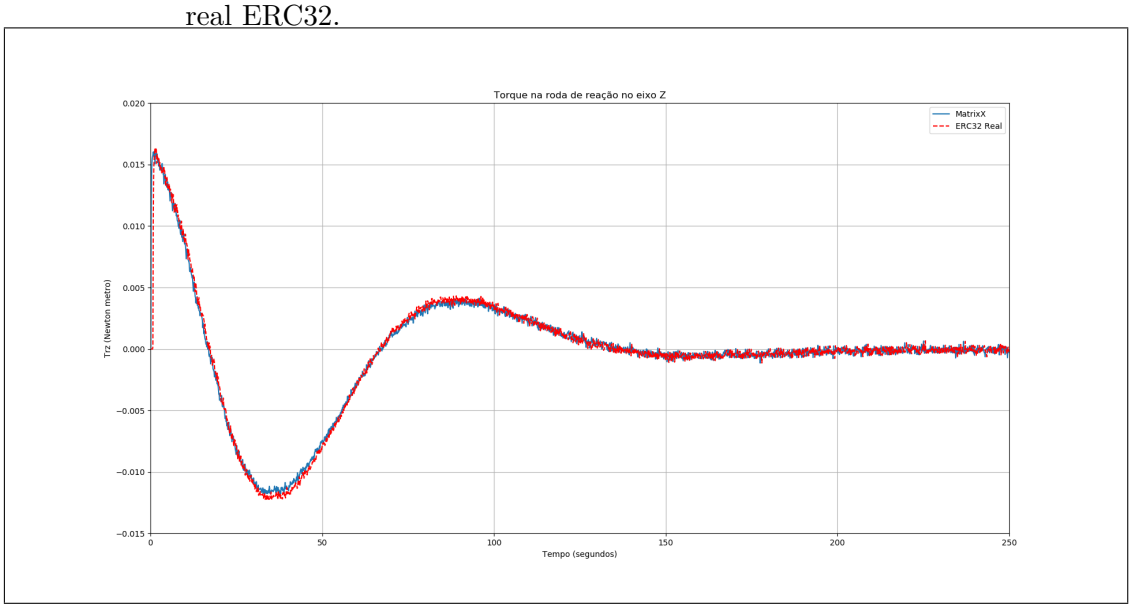
Fonte: Amorim III (2009).

Figura A.88 - Erro entre os valores do torque da roda de reação no eixo Z (Trz) para os casos: simulado no ambiente MatrixX/SystemBuild e no ambiente ERC32CCS-Linux Multiprocessado.



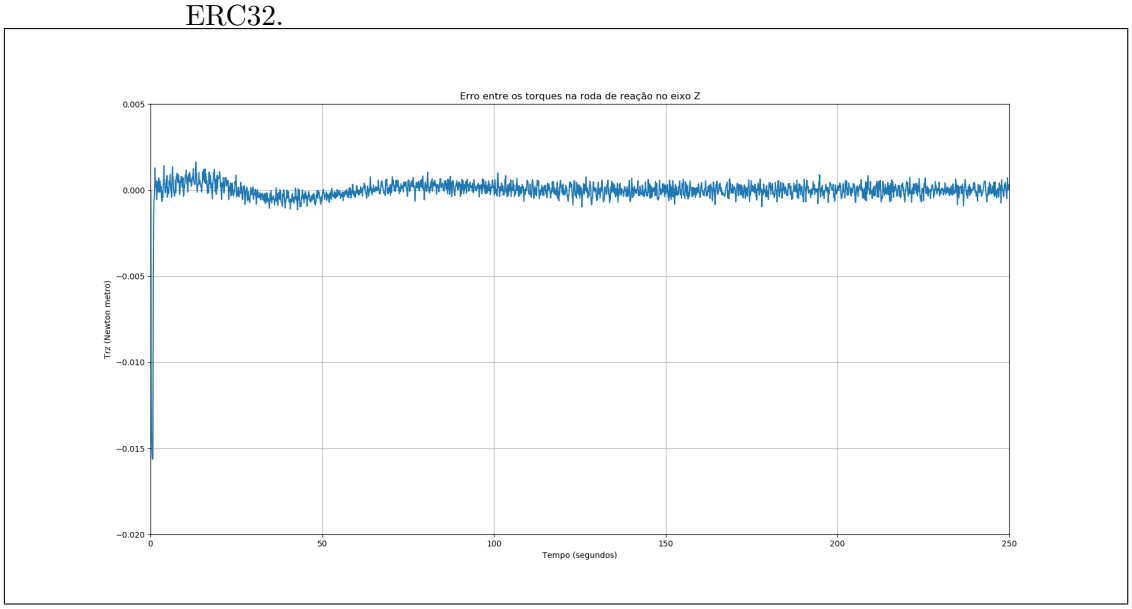
Fonte: Amorim III (2009).

Figura A.89 - Comparação entre os valores do torque da roda de reação no eixo Z (Trz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.

Figura A.90 - Erro entre os valores do torque da roda de reação no eixo Z (Trz) para os casos: simulado no ambiente MatrixX/SystemBuild e no processador real ERC32.



Fonte: O Autor.