UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA MECÂNICA

EDUARDO CAZARINI

Desenvolvimento de uma plataforma para testes de controladores, em arquitetura de controle *hardware in the loop*, utilizando um *hardware* eletrônico externo e um *software* de simulação de voo.

São Carlos

2014

EDUARDO CAZARINI

Desenvolvimento de uma plataforma para testes de controladores, em arquitetura de controle *hardware in the loop*, utilizando um *hardware* eletrônico externo e um *software* de simulação de voo.

Dissertação apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo para a Obtenção do Título de Mestre em Engenharia Mecânica.

Área de concentração: Dinâmica das máquinas e sistemas.

Orientador: Prof. Dr. Daniel Varela Magalhães

São Carlos

2014

ESTE EXEMPLAR TRATA-SE DA VERSÃO CORRIGIDA. A VERSÃO ORIGINAL ENCONTRA-SE DISPONÍVEL JUNTO AO DEPARTAMENTO DE ENGENHARIA MECÃNICA DA EESC-SP

C386d

Cazarini, Eduardo
Desenvolvimento de uma plataforma para testes de controladores, em arquitetura de controle hardware in the loop, utilizando um hardware eletrônico externo e um software de simulação de voo. / Eduardo Cazarini; orientador Dr. Daniel Varela Magalhães. São Carlos, 2015.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Mecânica e Área de Concentração em Dinâmica de Máquinas e Sistemas -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. Hardware in the loop. 2. Software in the loop. 3. Microsoft Flight Simulator. 4. Controlador H infinito. 5. Sistemas embarcados. 6. Quadricóptero. 7. dsPIC. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **EDUARDO CAZARINI.**

Título da dissertação: "Desenvolvimento de uma plataforma para testes de controladores, em arquitetura de controle hardware in the loop, utilizando um hardware eletrônico externo e um software de simulação de vôo".

Data da cefesa: 06/03/2015

Comissão Julgadora:	**Resultado:**
Prof. Dr. **Daniel Varela Magalhães** **(Orientador)** (Escola de Engenharia de São Carlos/EESC)	APROVADO
Prof. Dr. **Rafael Vidal Aroca** (Universidade Federal de São Carlos/UFSCar)	APROVADO
Prof. Dr. **Sergio Ricardo Muniz** (Instituto ce Física de São Carlos/IFSC)	Aprovado

Coordencdor do Programa de Pós-Graduação em Engenheira Mecânica:
Prof. Associado **Marcelo Areias Trindade**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

À minha filha Ana Alice.

AGRADECIMENTOS

Ao Prof. Dr. Daniel, que proporcionou dedicada e atenciosa orientação na condução deste trabalho.

À minha esposa Ana Claudia, pelo apoio constante, carinho e paciência ao longo de todas as etapas da minha vida.

Aos meus pais e familiares, que sempre confiaram em mim, me apoiaram e proporcionaram os valores que eu carrego hoje.

Aos amigos, em especial Rafael Coronel, André Carmona e Rômulo Rodrigues e aos professores Marcelo Becker e Adriano Siqueira pela considerável ajuda no desenvolvimento deste trabalho.

À Universidade de São Paulo, que me proporcionou oportunidades para uma formação de qualidade.

Aos docentes e funcionários da unidade da USP de São Carlos, que contribuem para a qualidade do ensino da Universidade.

Ao CNPQ pelo apoio financeiro.

RESUMO

CAZARINI, Eduardo. Desenvolvimento de uma plataforma para testes de controladores, em arquitetura de controle *hardware in the loop*, utilizando um *hardware* eletrônico externo e um *software* de simulação de voo. Dissertação de mestrado – Departamento de Engenharia Mecânica da Escola de Engenharia de São Carlos. São Carlos: Universidade de São Paulo, 2014.

Essa dissertação tem por objetivo o desenvolvimento de uma plataforma para testes de controladores de voo. Tal plataforma consiste em um *hardware* executando algoritmos de controle e atuando numa aeronave simulada em *software* de simulação de voo. O *software* de simulação escolhido, baseado na experiência prática de pilotos profissionais, foi o *Microsoft Flight Simulator* (MSFS), para o qual desenvolveu-se o modelo gráfico e dinâmico do quadricóptero AscTec Pelican. A comunicação entre o MSFS e o *hardware* é feita pela interface USB através do *software* FVMS v2.0 desenvolvido em ambiente DELPHI® 7.0 exclusivamente para este trabalho. O FVMS é capaz de ler o estado das variáveis de voo no MSFS, enviá-las para o *hardware* externo executar o controle, receber os sinais de controle de volta e utilizá-los no MSFS. O projeto e execução do *hardware* externo com controlador dsPIC também foi realizado neste mesmo trabalho. A título de avaliação de desempenho, também foi implementado um controlador robusto do tipo H∞ linear, desenvolvido pela equipe ART (*Aerial Robots Team*) da Escola de Engenharia de São Carlos. O mesmo controlador também foi aplicado na arquitetura *software in the loop*, na qual o controle é executado dentro do próprio FVMS, para comparação de desempenho entre os dois sistemas. Ao término do trabalho, as características de desempenho do sistema como um todo ficam bem evidenciadas através dos testes de estabilidade com e sem distúrbios executados em ambas arquiteturas de controle.

Palavras-chave: *Hardware in the loop*, *Software in the loop*, sistema microprocessado, controle H∞.

ABSTRACT

CAZARINI, Eduardo. *Development of a platform for controllers tests, in hardware in the loop control architecture, using an external electronic hardware and a flight simulation software. Dissertation – School of Engineering os Sao Carlos, University of Sao Paulo, 2014.*

This dissertation aims to develop a platform for flight controllers tests. It platform consists of an electronic hardware where the control´s algorithms will be executed and a virtual aircraft is simulated in flight simulation software. The chosen simulation software, based on practical experience of professional pilots, was Microsoft Flight Simulator (MSFS). The graphic and dynamic model of quadrotor AscTec Pelican was developed to perform inside the software. The communication between the MSFS and the hardware is made by USB interface through FVMS v2.0 software developed in DELPHI® 7.0 environment, exclusively for this work. The FVMS can read the status of the flight variables in MSFS, send them to the external hardware, receive control signals back and write them in MSFS. The design and implementation of external hardware with dsPIC controller was also developed ons ame work. For performance evaluation of the system, it was also implemented a robust linear H∞ controller, developed by ART team (Aerial Robots Team) of the School of Engineering of São Carlos. The same controller was also applied using software in the loop architecture, in which the control is performed inside FVMS, to compare performance between the two architectures. In the end of the work, the performance characteristics of the systems were well evidenced by the stability tests carried out with and without disturbances in both control architectures.

Keywords: Hardware in the loop, Software in the loop, embedded systems, H∞ control.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1.Introdução

1.1. Justificativa

Num passado não muito distante, os métodos tradicionais de desenvolvimento de produtos diziam que, em linhas gerais, o produto deveria ser desenvolvido e testado em protótipos físicos, semelhantes ao produto final. Essa metodologia foi e ainda é aplicada por muitas empresas no BraSIL ou mesmo em países do primeiro mundo. No entanto, existem dois problemas relacionado à ela: alto tempo de desenvolvimento e alto custo (ROZENFELD *et al.*, 2006).

Hoje em dia, em um mercado altamente competitivo, as empresas não podem se dar ao luxo de atrasar seus desenvolvimentos e de comprometer seu orçamento durante o desenvolvimento do produto, sob pena, inclusive, de perder mercado para seus concorrentes. Pensando nisso, novas metodologias para o desenvolvimento de produtos têm usado a tecnologia para propor formas mais eficientes de gerar produtos cada vez melhores, mais baratos e mais rapidamente.

O uso de simulação durante as fases do desenvolvimento do produto é uma solução para essas duas questões mencionadas nos parágrafos anteriores. A simulação do produto em sistemas computacionais permite que os engenheiros interajam com o produto virtualmente, sem que ele exista, e depois construam o protótipo físico mais próximo do produto ideal (ROZENFELD *et al.*, 2006). E é essa linha de raciocínio que guia as novas metodologias de desenvolvimento de produtos. Alguns exemplos de *software*s de simulação podem ser citados: elementos finitos, simulação numérica com MATLAB®/Simulink, sistemas CAD/CAM/CAE e, porque não, simuladores de voo.

No mercado aeronáutico esses problemas se agravam. Os custos relacionados a esse tipo de produto são altos e além disso, os testes em protótipos físicos geralmente são difíceis de serem feitos, pois muitos deles são feitos durante o voo. Também há a questão das vidas humanas envolvidas nesses ensaios.

O trabalho desenvolvido nesta dissertação caminha justamente nesse sentido. Testar o *hardware* externo num ambiente virtual de simulação que transmita o máximo

de realismo possível, facilita consideravelmente os testes dos elementos que, eventualmente, poderiam ser inseridos num protótipo real da aeronave.

1.2. Objetivo

O objetivo deste trabalho é construir uma plataforma eletrônica que possa ser utilizada para testar controladores que objetivam o controle de estabilidade de um quadricóptero em malha fechada, utilizando um *software* para PC como simulador de voo. O quadricóptero tem sua dinâmica simulada no *software Microsoft Flight Simulator* e é controlado pelo sistema embarcado. A comunicação com o sistema é feita via comunicação serial USB. Para avaliar o sistema, também foi implementado um controle robusto H_∞. Outros tipos de controladores podem ser testados nessa plataforma, mas o uso de um controlador H_∞ trás mais inovação ao projeto, além de dar continuidade aos projetos desenvolvidos pela equipe ART (*Aerial Robots Team*). Para avaliar as qualidades da simulação na arquitetura *hardware in the loop* (HIL), os mesmos testes são executados usando a arquitetura *software in the loop* (SIL).

1.3. Método

A primeira fase da pesquisa é constituída de uma pesquisa bibliográfica sobre os assuntos abordados no trabalho, tais como, arquiteturas de controle (*software in the loop*, *man in the loop* e *hardware in the loop*), simuladores de voo, modelagem e controle do quadricóptero Pelican AscTec.

Na segunda fase o *hardware* para controle foi desenvolvido, bem como sua respectiva programação. O *hardware* desenvolvido possui outras funções não utilizadas nesse trabalho, mas que serão utilizados em trabalhos futuros.

Na terceira fase, foi realizada a programação do *software* FVMS V2.0 para integração do *hardware* com o *software* de simulação *Microsoft Flight Simulator* (MSFS). Um controle robusto do tipo H_∞ foi implementado para que se possa avaliar o comportamento do sistema como um todo. Para vias de comparação, o mesmo sistema também foi programado para trabalhar na arquitetura SIL (*software in the loop*), na qual o controle foi realizado diretamente pelo FVMS, sem o uso do *hardware* externo.

Por fim, alguns testes de desempenho foram realizados e comparados entre as duas arquiteturas de controle, *software in the loop* e *hardware in the loop*. Esperava-se que os resultados do SIL fossem ligeiramente melhores do que os do HIL, pois o tempo de comunicação entre o *hardware* e o computador resulta em períodos maiores de processamento e, consequentemente, em frequências menores de atualização dos atuadores de controle. No entanto, o que se observou foi que este tempo de comunicação pouco impactou nos resultados em comparação ao tempo do processamento de imagens do MSFS pelo sistema operacional.

1.4. Arquitetura de sistemas de controle

Entende-se como controlar o ato de alterar a resposta transitória e estacionária de uma planta a um determinado estímulo externo. A Figura 1 mostra esquematicamente um sistema de controle padrão.

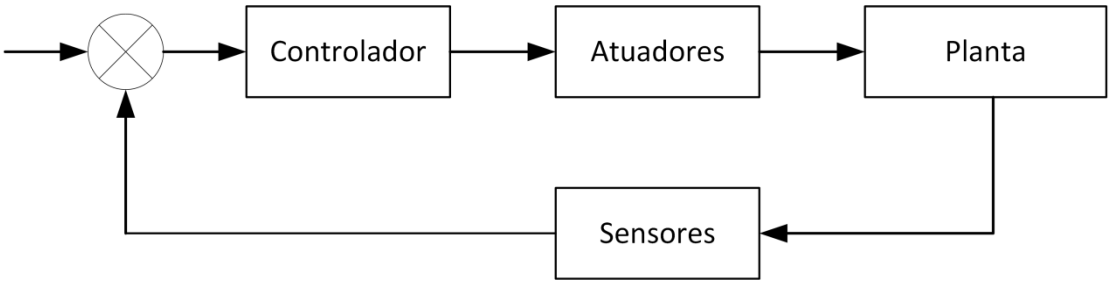


Figura 1-Sistema de controle padrão

No caso deste trabalho, a planta é o quadricóptero Pelican, que é simulado no MSFS. Usualmente, existem 3 arquiteturas básicas de controle utilizadas em simulações durante o projeto de sistemas mecatrônicos:

1. *Software in the loop* (SIL): Neste caso, o controlador é simulado por *software*, na mesma máquina, ou em mais do que uma com comunicação por rede, onde o MSFS simula o voo com a aeronave. Esse tipo de arquitetura é utilizado num estágio inicial do desenvolvimento no qual o *hardware* real de controle ainda não existe, mas os testes de funcionamento do controlador são desejados;
2. *Man in the loop*: Neste caso, o controlador é a pessoa como, por exemplo, o jogador do MSFS que controla a aeronave através do *joystick,* ou a pessoa que pilota o quadricóptero utilizando o controle remoto num sistema real. No caso de

um aeromodelo real, o piloto seria a pessoa que controla o avião pelo controle remoto por rádio frequência;

3. *Hardware in the loop* (HIL): No *hardware in the loop*, o controlador deve ser colocado em um *hardware* externo que se comunica com a planta por algum protocolo de comunicação específico. Esse tipo de arquitetura é utilizado nos estágios intermediários do desenvolvimento no qual o *hardware* de controle já existe;

Nesta dissertação, a arquitetura de controle utilizada foi a *hadware in the loop(HIL)*, porém, para testes de performance, também foi utilizada a arquitetura *software in the loop*.

1.5. Estrutura do trabalho

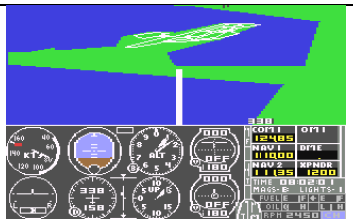
No capítulo 2 são apresentados os conceitos de simuladores de voo, especialmente o *Microsoft Flight Simulator* (MSFS). Lá também é justificada a escolha do MSFS. No capítulo 3, são apresentados a modelagem dinâmica do quadricóptero ASCTEC PELICAN e o algoritmo de controle H_∞ usado neste trabalho. No capítulo 4, é descrito todo o desenvolvimento do *hardware* eletrônico desenvolvido e do software de apoio FVMS. No capítulo 5 são apresentados os testes de estabilidade realizados bem como seus respectivos resultados. Por fim, o capítulo 6 conclui essa dissertação, com uma discussão sobre o que esperar do sistema quando o *hardware* for embarcado em um protótipo real.

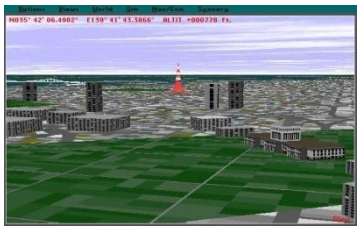
2.Simuladores de voo

Para que as técnicas de desenvolvimento de sistemas utilizando as arquiteturas *software in the loop* (SIL) ou *hardware in the loop* (HIL) possam ser executadas, deve-se escolher com muito cuidado o simulador virtual da planta que se deseja testar. Este capítulo tem por objetivo apresentar e justificar a escolha do *software Microsoft Flight Simulator* (MSFS) usado neste trabalho.

O *Microsoft Flight Simulator* (MSFS) é um *software* para simulação de aeronaves em voo, desenvolvido e comercializado pela Microsoft. Foi inicialmente desenvolvido por Bruce Artwick na empresa SUBLOGIC em 1977, passou por algumas versões específicas para algumas plataformas de *hardware*, até chegar na primeira versão para IBM-PC comercializada pela Microsoft em 1982. De 1982 até 2014 foram lançadas mais 9 versões, até chegar nas versões finais FS2004 e FSX, de 2003 e 2006 respectivamente. A evolução mais notável entre essas versões está na interface gráfica que acompanhou a evolução do *hardware* durante esse mesmo período. Com o passar do tempo, o programa também ganhou novos recursos de iteração entre o usuário, o avião pilotado e o ambiente onde ele se encontra. A Tabela 1 (História do Flight Simulator, 2014) mostra alguns exemplos dos recursos adquiridos em cada versão do programa.

Tabela 1- Evolução do Microsoft Flight Simulator
Fonte: História do Flight Simulator (2014)

Versão	Ano	Características	Imagens
Flight Simulator 1.0	1982	Interface gráfica pobre, compatível com o contexto do *hardware* da época.	
Flight Simulator 2.0	1984	Adição de joystick e mouse. Pouca melhoria nos gráficos.	

Flight Simulator 3.0	1988	Apresentava diferentes exibições móveis em 3D da janela e do painel.	
Flight Simulator 4.0	1989	Introdução de estradas, pontes e edificações com algum nível de detalhe. Passou a permitir que os usuários projetassem suas aeronaves, o que possibilitou o surgimento de outros produtos agregados ao MSFS.	
Flight Simulator 5.0	1993	Cenários foto-realísticos e um banco de dados do mundo inteiro renderizados com imagens em 3D, resolução super VGA e 256 cores. A vista da cabine de comandos passou a se parecer com as cabines reais.	
Flight Simulator 5.1	1995	Bibliotecas de cenários e efeitos meteorológicos (chuva, tempestades, neblina, etc). São elementos que melhoram o realismo da simulação.	
Flight Simulator 95	1996	Melhorias em geral nos gráficos além da inclusão de novas aeronaves e de painéis cada vez mais realísticos. Também foram incorporados efeitos de iluminação, sombras, objetos celestes e meteorológicos. O produto incorporou também aulas de aviação e facilitou o processo de instalação, visando ampliar o leque de clientes.	

Flight Simulator 98	1997	Número de aeroportos 10 vezes maior que o da versão anterior. Suporte à placas aceleradoras 3D. Aumento do número de aeronaves. Aperfeiçoamento geral das imagens gráficas.	
Flight Simulator 2000	1999	Aumento do número de aeronaves e de aeroportos. Cidades mais detalhadas com imagens de maior precisão, além da maior quantidade de dados atualizados sobre navegação e meteorologia.	
Flight Simulator 2002	2001	Instrumentos funcionais no cockpit e vistas de câmeras em movimento, trazendo uma sensação ao usuário ainda mais realística. As edificações, a vegetação e a topografia passaram a ser totalmente fiéis às áreas sobrevoadas.	
Flight Simulator 2004	2003	Imprevisibilidade do tempo e vários outros fatores aleatórios são extraídos da vida real. Aumento da interatividade dos cockpits. Também foram implementados recursos como controle terrestre e meteorológico.	
Flight Simulator FSX	2006	Aprimoramento ainda maior das interfaces gráficas. Possibilidade de ser copiloto na aeronave de outra pessoa na função *multi-player*.	

Uma característica importante do MSFS é a possibilidade de customização ou mesmo inserção de novas aeronaves no programa. Isso permite aos usuários interações com diversos recursos do programa, tais como: o modelo da aeronave, a imagem e o *layout* do *cockpit*, as características de voo, modelos, *layouts* e texturas de cenários, entre outros. O modelo de uma aeronave é feito basicamente por 5 partes:

- Modelo: desenho em CAD 3D. Referem-se aos desenhos externos e internos da aeronave;
- Texturas: arquivo bmp com a figura do que vai impresso na fuselagem dos aviões;
- Sons: Arquivos WAV com os sons da aeronave;
- Painel: bitmaps com os desenhos do painel da aeronave;
- FDE (*Flight Dynamics Engine*): Constituído por 2 arquivos: **airfile.air**, arquivo que contém centenas de parâmetros que definem as características de voo da aeronave e o **aircraft.cfg**, constituído por alguns outros parâmetros de fácil alteração.

O MSFS foi desenvolvido inicialmente com o intuito de ser apenas um jogo com arquitetura de controle do tipo MIL (*Man in the loop*), ou seja, aquela em que o controle é feito pelo homem com o uso do *joystick*. Por isso, o MSFS não é muito utilizado em artigos acadêmicos, quando comparado com os simuladores *X-Plane* e *FlightGear*. Um outro problema do MSFS é que a Microsoft não revela detalhes técnicos sobre os modelos dinâmicos dos aviões.

No entanto, segundo Louali et al. (2011) o MSFS reproduz melhor as reais sensações referentes ao voo do que seus concorrentes diretos. O MSFS utiliza tabelas de parametrização do modelo enquanto que o *X-Plane* utiliza um método chamado *"blade element modeling"*. Segundo os autores deste trabalho, a fidelidade da simulação depende mais da acurácia do modelo do que do método de modelagem utilizado.

Para este trabalho especificamente, as sensações reais não fazem muita diferença, e o uso de um modelos mais preciso seria, inclusive, mas interessante. No entanto, outras pesquisas, nas quais essas sensações são importantes, estão sendo desenvolvidas pelo mesmo grupo de pesquisa ART. Essas pesquisas utilizarão este trabalho como base.

Com base nos trabalhos de Louali et al. (2011) e Cantoni et al (2009), segue abaixo os motivos pela escolha do *Microsoft Flight Simulator 2004* como simulador de voo para este trabalho:

- Trabalha em tempo real, (LOUALI, *et al.*, 2011) e (FELDSTEIN e MUZIO, 2004);
- O MSFS reproduz melhor as sensações durante as simulações do que seus concorrentes diretos *X-Plane* e *FlightGear*, conforme demonstrado por Louali et al. (2011). Em forums específicos sobre simuladores de voo, também é possível verificar essa preferência dos usuários;
- O MSFS oferece uma interface gráfica de alta qualidade, o que permite a visualização em tempo real do comportamento da aeronave em voo, (LOUALI, *et al.*, 2011). Este recurso é particularmente importante quando se deseja aferir ou ajustar os parâmetros do controlador tomando como base as reações da aeronave visíveis na interface gráfica do software;
- FS2004 possui uma rica biblioteca de cenários e aeronaves, além de permitir a criação de aeronaves customizáveis. Este recurso foi especialmente importante neste trabalho pois possibilitou a inserção do quadricóptero no simulador;
- O FS2004 ainda possui APIs desenvolvidas pela Microsoft ou por terceiros, que permitem a interação direta com o modelo simulado por linhas de comando. Este recurso possibilitou a criação do *software* FVMS V2.0 utilizado como interface entre o MSFS e o *hardware* externo através do uso da *DLL* FSUIPC, desenvolvida por Peter Dowson (DOWSON, 2014).
- O MSFS exige pouco recurso computacional se comparado com seus concorrentes diretos, além de ser considerado de baixo custo. A Tabela 2 compara a versão FS2004 com o *X-Plane* e a versão FSX, considerando alguns parâmetros de desempenho (LOUALI *et al.*, 2011). O *FlightGear* é um simulador de voo bastante utilizado em pesquisas como esta. Além de possuir praticamente as mesmas qualidades do MSFS, ainda tem a seu favor o fato de ser gratuito e de código aberto. A
- Tabela 3 e a Figura 2 mostram alguns resultados comparativos entre o *FlightGear* e o MSFS FSX, obtidos no trabalho de Cantoni et al. (2009). Apesar disso, justamente por haver poucos trabalhos acadêmicos com ele e pelo fato deste

trabalho estar inserido numa linha de pesquisa já existente no ART, o MSFS acabou parecendo a melhor opção.

Tabela 2 - Comparação entre FSX, X-Plane e FS2004.
Fonte: Louali et al. (2011).

Features	FSX	X-plane9	FS2004
Modelo Dinâmico	Paramétrico	*Blade element teory*	Paramétrico
O.S.	Windows Vista®	Windows 7®	Windows XP®
Processor	1.0GHz	1.0GHz	450MHz
RAM	512MB	1GB	128MB
Hard Disc	14GB	6-60GB	1,8GB
Video Card	32MB	64MB	8MB
Price	€44.00	€59.99	€14.00

Tabela 3 - Comparação FlightGear X MSFS FSX.
Fonte: Adaptado de Cantoni, et al (2009).

Característica	**FlightGear**	**MSFS FSX**
Custo da ferramenta	Baixo	Médio
Facilidade de utilização do simulador na integração *hardware-in-the-loop*	Alta	Baixa
Portabilidade	Alta	Baixa
Qualidade gráfica da simulação	Baixa	Alta
Frequência de simulação	Alta	Média
Construção de aeronaves	Baixa	Média
Simulação multi-veículos	Baixa	Baixa
Extração de imagem para controle servo-visual	Baixa	Baixa
Documentação	Alta	Média

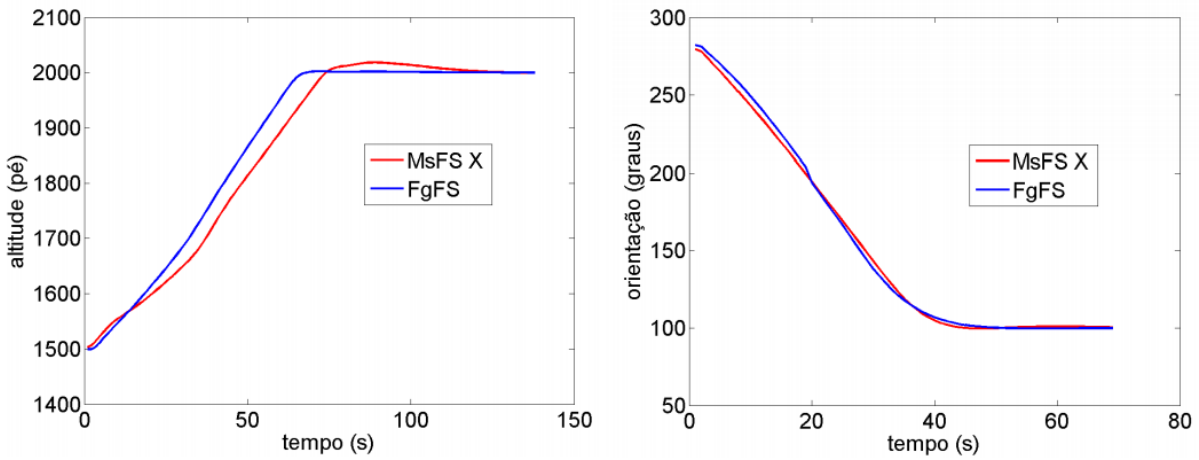


Figura 2 - Controle de altitude (figura da esquerda) e orientação (figura da direita) com controlador PID com FlighGear e o MFSF FSX.
Fonte: Cantoni, et al (2009).

3. Modelagem e controle do quadricóptero

Este trabalho descreve o desenvolvimento de uma plataforma para testes de controladores, utilizando um *hardware* externo na execução do controle e o *Microsoft Flight Simulator* como simulador de voo das aeronaves. Para que a simulação ocorra, deve-se escolher uma aeronave de interesse, determinar sua modelagem dinâmica e definir os respectivos coeficientes das equações geradas. Para o caso em questão, escolheu-se como aeronave o helicóptero do tipo quadricóptero PELICAN disponível no laboratório de robótica aérea móvel da Escola de Engenharia de São Carlos (ART).

O PELICAN é fabricado pela empresa alemã ASCTec e foi adquirido por se tratar de uma plataforma que dispõe de grande quantidade de recursos eletrônicos de localização e comunicação. A aquisição foi realizada em 2013 e, desde então, diversos trabalhos de pesquisa foram realizados. Dentre esses trabalhos, alguns são relativos a controladores robustos lineares do tipo H_∞. Este tipo de controlador é relativamente novo e se trata de uma opção aos tradicionais controladores PID, largamente utilizados em trabalhos acadêmicos e, principalmente, em projetos industriais.

As duas próximas seções descrevem a modelagem do quadricóptero e as características do controlador H_∞ utilizados no desenvolvimento dessa dissertação.

3.1. Modelagem do Quadricóptero Pelican

O helicóptero do tipo quadricóptero consiste num modelo de aeronave que possui 4 rotores verticais diametralmente opostos. Este conceito de aeronave surgiu no início do século passado quando, em 1907, os irmãos Bréguet orientados pelo professor Richet desenvolveram uma aeronave batizada de Bréguet-Richet *Gyroplane No 1* (Figura 3). Tratava-se de um helicóptero grande, com um motor central de combustão interna que transmitia o movimento para os 4 rotores, com 8,1 m de diâmetro cada, através de polias e correias. O piloto ficava na região central, acima do motor que gerava até 40hp de potência. A estrutura do helicóptero pesava 345 kg, que somados aos 165 kg no motor e aos 68 kg do piloto de teste, resultaram num conjunto com massa total de 578 kg, (LEISHMAN, 2006).

No primeiro experimento realizado pelos irmãos, relata-se que o helicóptero saiu do chão mas não conseguiu subir mais do que 1,5m de altura, graças a seu elevado peso e à falta de controlabilidade da aeronave. O único controle que o piloto tinha era o da aceleração vertical da mesma, entretanto este tipo de aeronave é extremamente instável. Controlá-la é semelhante à controlar um pêndulo invertido, impossível para a época em que este experimento fora realizado.

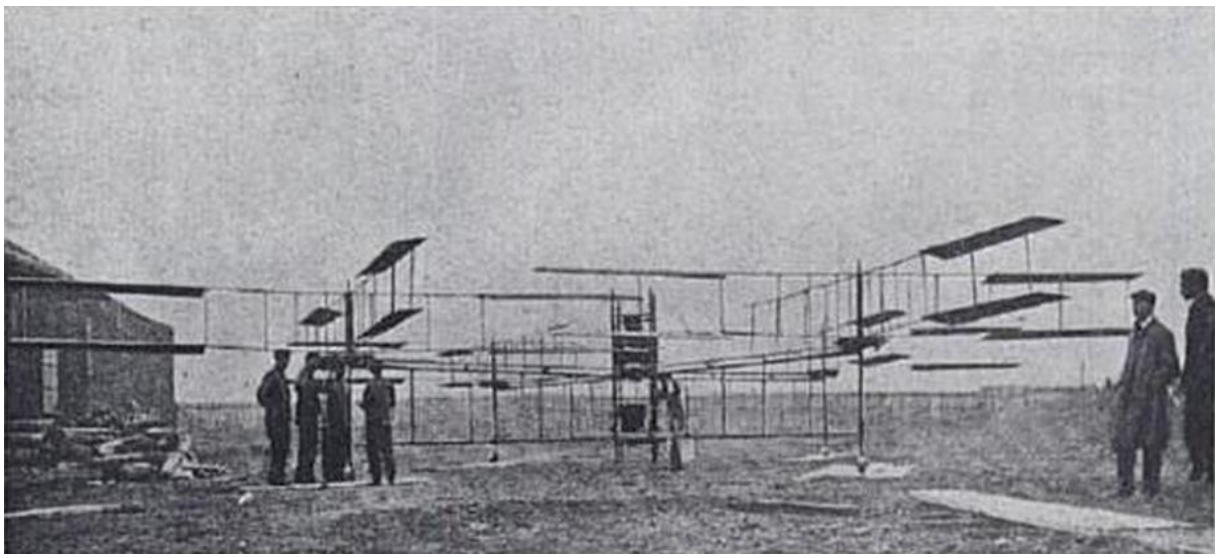


Figura 3 - Gyroplane No. 1.
Fonte: Leishman (2006).

Desde então, outros experimentos existiram e recentemente os quadricópteros voltaram a ser objeto de pesquisa por se tratarem de uma boa opção para veículos aéreos não tripuláveis de pequeno porte, VANT, MAV ou AVG.

A Figura 4 mostra uma classificação de veículos aéreos não tripulados sugerida por Siegwart, Nourbakhsh e Scaramuzza (2011). Observe que o quadricóptero é considerado pelos autores como sendo uma aeronave mais pesada que o ar, motorizada, com asas rotativas e do tipo VTOL (*Vertical Take-Off and Landing*).

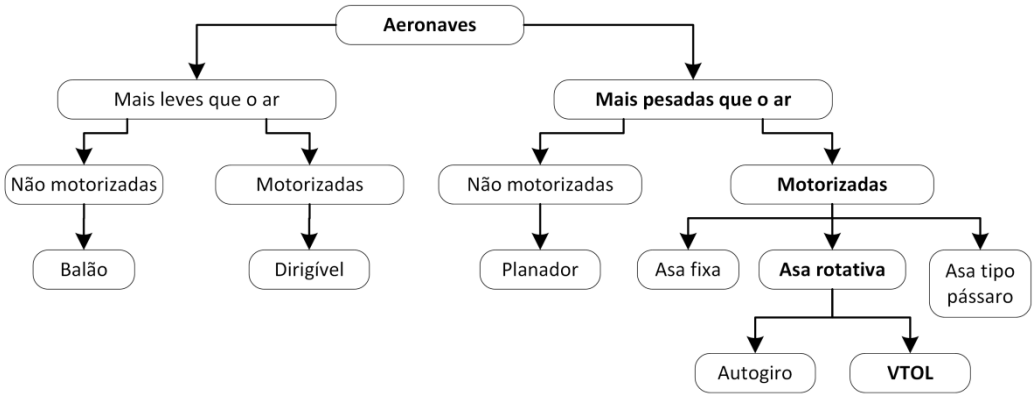


Figura 4 - Classificação de aeronaves.
Adaptado de Siegwart, Nourbakhsh e Scaramuzza (2011).

Entre os diferentes tipos de veículos aéreos, uma característica específica dos modelos com asas rotativas os colocam em uma posição diferenciada. Eles são os únicos que conseguem levantar voo na vertical, ficar estáticos (*hovering*) e se deslocar em baixas velocidades. Os modelos VTOL, especificamente, apesar de serem difíceis de serem controlados, possuem características construtivas simples e flexíveis, o que os torna uma solução atrativa para desenvolvimentos de trabalhos acadêmicos.

Até 2006, o principal problema para os pesquisadores estava na estabilização do quadricóptero, especialmente nos modelos menores. Desde 2007, o foco das pesquisas mudaram para a navegação autônoma, inicialmente *outdoor* e, mas recentemente, *indoor* (SIEGWART, NOURBAKHSH e SCARAMUZZA, 2011). O avanço da tecnologia de micro sensores (MEMS), que estão cada vez menores e mais precisos, aliado à respectiva queda nos custos de aquisição, impulsionam ainda mais essas pesquisas. A Tabela 4 relaciona algumas vantagens e desvantagens dos quadricópteros.

Tabela 4- Vantagens e desvantagens dos quadricópteros.
Fonte: Bouabdallah, Murrieri e Siegwart (2004).

Vantagens	Desvantagens
Mecânica simples	Tamanho relativamente grande
Altos níveis de payload	Pesados
Efeito giroscópio reduzido	Alto consumo de energia

Entre as desvantagens, destaca-se o peso e o alto consumo de energia. As pesquisas realizadas pelo ART indicam que a redução do peso impacta no respectivo consumo de energia. O uso de materiais resistentes porém leves, como a fibra de carbono, vem sendo recomendado para este tipo de aplicação. Com relação ao consumo, existem pesquisas em andamento que tentam mesclar aeronaves do tipo VTOL com aeronaves do tipo planador, que permitam o voo em determinadas situações, sem a necessidade dos motores (SAMPAIO *et al.*, 2014).

3.1.1. Definição dos vetores de rotação e translação do quadricóptero

A Figura 5 foi feita pelo ART para definir o sentido dos vetores de rotação e translação do quadricóptero Pelican. Esta figura, portanto, é o ponto inicial da modelagem utilizada neste trabalho. Observe o sentido de giro dos rotores. Sempre que uma hélice gira em torno do eixo vertical, junto com a propulsão e a sustentação, aparece também um torque contrário ao sentido de giro. Isso acontece devido à terceira lei de Newton, ação e reação, e se não for compensada, resulta no giro do helicóptero em sentido oposto. A oposição nos sentidos de giro entre os motores pares e os ímpares é necessária para que o torque gerado por um dos pares seja anulado pelo torque gerado pelo outro.

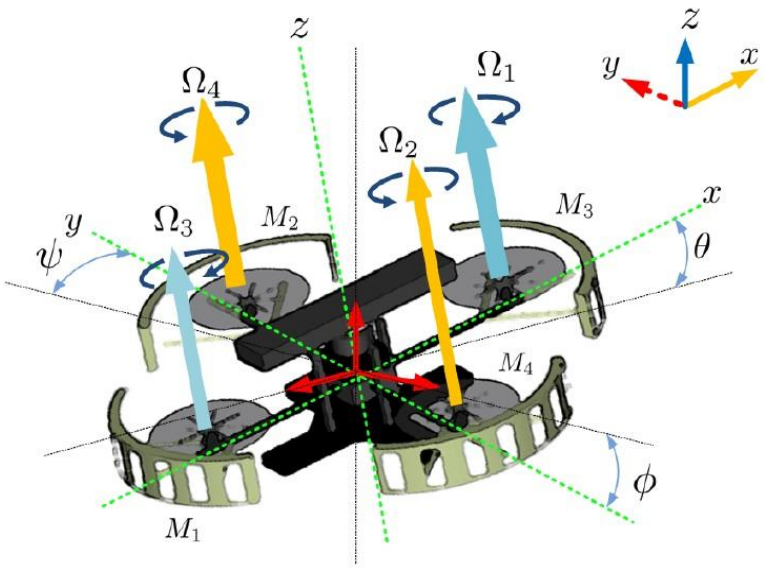


Figura 5 - Definição dos vetores de rotação e translação do Pelican
Fonte: Sampaio et al. (2013b).

Para este trabalho, consideraremos θ como sendo o ângulo de arfagem (*pitch*), ϕ como sendo a rolagem (*roll*) e ψ o ângulo de guinada (*yaw*). A alteração nas velocidades dos rotores impacta diretamente no controle de atitude do helicóptero e, consequentemente, no seu deslocamento no espaço. A Figura 6 e a Figura 7 ilustram alguns movimentos relevantes consequentes das alterações dessas velocidades.

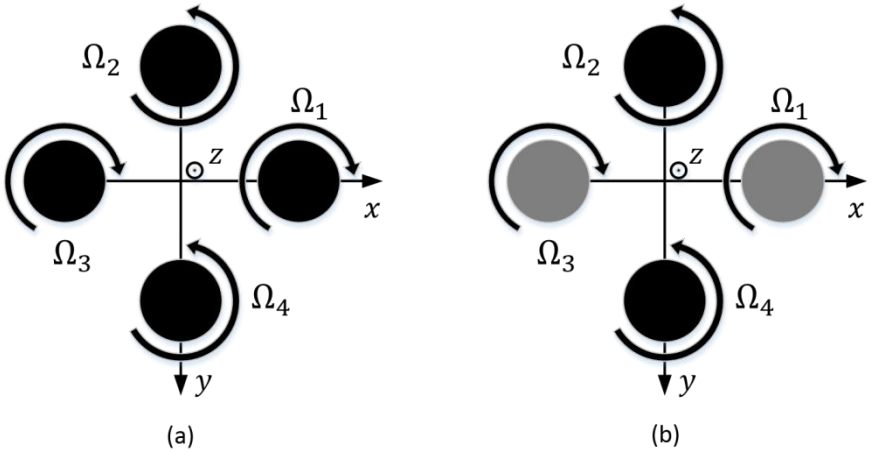


Figura 6 - Incremento e decremento de velocidades nos rotores nos movimentos vertical (a) e guinada (b)

Para que haja deslocamento na vertical (z), deve-se aumentar ou diminuir as velocidades de forma proporcional em todos os motores (Figura 6a). A alteração no angulo de guinada (ψ) pode ser realizada colocando-se velocidades diferentes para os dois conjuntos de rotores, ou seja Ω_1=Ω_3 e Ω_2=Ω_4 (Figura 6b).

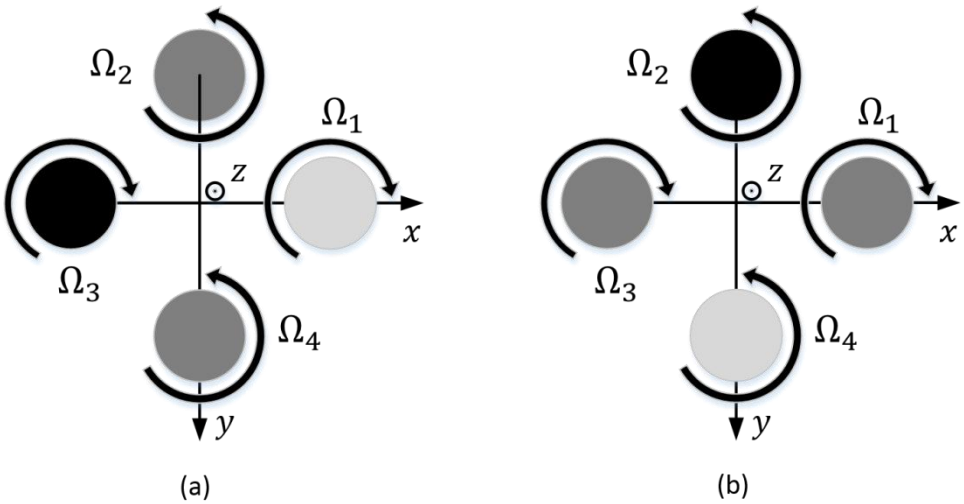


Figura 7 - Incremento e decremento de velocidades nos rotores nos movimentos arfagem (a) e rolagem (b)

Para executar alteração no ângulo de arfagem (θ), deve-se manter a rotação dos motores Ω_2 e Ω_4 e alterar as rotações dos motores Ω_1 e Ω_3 (Figura 7a). Este mesmo procedimento também gera translação em x. Para executar alteração na rolagem (ϕ), deve-se manter a rotação dos motores Ω_1 e Ω_3 e alterar as rotações dos motores Ω_2 e Ω_4 (Figura 7b). Este mesmo procedimento também gera translação em y.

3.1.2. Considerações gerais

A modelagem matemática do quadricóptero já foi publicada em diversos trabalhos acadêmicos nos últimos anos. Está muito bem detalhada nos trabalhos de Bouabdallah, Murrieri e Siegwart (2004) e Mahony, Kumar e Corke (2012). Também foi mencionada nos trabalhos de Sampaio et al (2013a), Sampaio et al (2013b), Sampaio et al (2013c).

O desenvolvimento da modelagem do quadricóptero foi realizado com as seguintes considerações:

- A estrutura da aeronave é considerada como um corpo rígido;
- A estrutura da aeronave é considerada simétrica;
- O centro de gravidade da aeronave é coincidente com o seu eixo de coordenadas local;
- As hélices são rígidas e não sofrem deformação;
- A força de propulsão e o arrasto são proporcionais ao quadrado da velocidade das hélices.

A Tabela 5 relaciona os principais efeitos naturais atuantes no helicóptero.

Tabela 5 - Efeitos naturais atuantes no quadricóptero.
Fonte: Bouabdallah, Murrieri e Siegwart (2004).

Efeito	Fonte
Efeitos aerodinâmicos	Rotação das hélices Assimetria no movimento das hélices
Torques inerciais	Mudança de velocidade das hélices
Gravidade	Posição do centro de massa
Efeito giroscópio	Mudanças de orientação do corpo rígido Mudanças de orientação do plano das hélices
Fricção	Movimento geral do helicóptero

3.1.3. Equações da modelagem dinâmica

Levando em conta as considerações da seção anterior, a execução da modelagem realizada por Bouabdallah, Murrieri e Siegwart (2004), nos retorna o conjunto de equações relacionadas em (1) e (2):

$$
\begin{aligned}
\ddot{x} &= \frac{1}{m}(cos\phi sen\theta cos\psi + sin\phi sen\psi)U_1 \\
\ddot{y} &= \frac{1}{m}(cos\phi sen\theta cos\psi - sin\phi sen\psi)U_1 \\
\ddot{z} &= -\mathrm{g} + \frac{1}{m}(cos\phi cos\theta)U_1 \\
\ddot{\phi} &= \left[\frac{I_y - I_z}{I_x}\right]\dot{\theta}\dot{\psi} - \frac{J_r\dot{\theta}\Omega}{I_x} + \frac{lU_2}{I_x} \\
\ddot{\theta} &= \left[\frac{I_z - I_x}{I_y}\right]\dot{\phi}\dot{\psi} - \frac{J_r\dot{\phi}\Omega}{I_y} + \frac{lU_3}{I_y} \\
\ddot{\psi} &= \left[\frac{I_x - I_y}{I_z}\right]\dot{\phi}\dot{\theta} - \frac{1}{I_z}U_4
\end{aligned}
\tag{1}
$$

onde,

- x, y, z : Deslocamentos em x, y e z;
- $\ddot{x}, \ddot{y}, \ddot{z}$: Acelerações em x, y e z;
- I_x, I_y, I_z: Momento de inércia em torno dos eixos x, y e z;
- ϕ, θ, ψ: Deslocamentos angulares em *roll, pitch e yaw*;
- $\dot{\phi}, \dot{\theta}, \dot{\psi}$: Velocidades angulares em *roll, pitch e yaw*;
- $\ddot{\phi}, \ddot{\theta}, \ddot{\psi}$: Acelerações angulares em *roll, pitch e yaw*;
- J_r: Momento de inércia dos rotores;
- l: Distância entre o centro do helicóptero e cada um dos rotores;
- U_1, U_2, U_3, U_4: Entradas do sistema, dadas pelas equações a seguir.

$$
\begin{aligned}
U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U_2 &= b(\Omega_4^2 - \Omega_2^2) \\
U_3 &= b(\Omega_3^2 - \Omega_1^2) \\
U_4 &= d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2)
\end{aligned}
\tag{2}
$$

onde

- $\Omega_{1...4}$ referem-se às velocidades dos rotores.
- b:coeficiente de aceleração;
- d:coeficiente de arrasto.

As incógnitas definidas nas equações (1) e (2) dependem das características construtivas da aeronave utilizada como modelo, apresentadas na Tabela 6. A maioria deles foi estimada pelo ART antes da compra dos quadricópteros. Recentemente, no entanto, um desenho mais completo em Solidworks® e uma simulação em Adams® foram realizados pelo grupo e os valores verificados. Alguns deles também foram obtidos no site do fabricante AscTec, (Asctec). De qualquer forma, esses dados entraram na síntese do controlador robusto $H\infty$, e uma das principais características do $H\infty$ é justamente o fato do modelo não precisar ser muito fiel à realidade.

Tabela 6 - Parâmetros estimados do Pelican.
Fonte: Sampaio et al. (2013c).

Variável	Valor	Unidade
Massa Total (m)	1,5	Kg
Coeficiente de aceleração (b)	0,4 - 0,6	-
Coeficiente de arrasto (d)	0,2 - 0,35	-
Distância do centro do quadricóptero até os rotores (l)	0,35	m
Momento de inércia I_x	$3{,}4\times10^{-3}$	$Kg.m^2$
Momento de inércia I_y	$3{,}4\times10^{-3}$	$Kg.m^2$
Momento de inércia I_z	$4{,}7\times10^{-3}$	$Kg.m^2$
Momento de inércia dos rotores (J_r)	$3{,}612\times10^{-5}$	$Kg.m^2$

Por fim, ainda tem a equação (3) que representa a função de transferência de primeira ordem, rotação dividida pelo sinal de controle, dos motores *Brushless* utilizados no PELICAN (SAMPAIO *et al.*, 2013a).

$$M(s) = \frac{0{,}316}{0{,}419s + 1{,}52} \tag{3}$$

3.2. Controle Robusto Linear H∞

A primeira questão a ser discutida nesta seção é a definição do termo "Robustez", principalmente no contexto de controle de sistemas. Um produto é considerado robusto quando, ao enfrentar problemas pelos quais não tenha sido projetado, ele consegue obter um desempenho satisfatório (ROZENFELD *et al.*, 2006). No contexto do controle de sistemas, a propriedade robustez está associada com a capacidade do sistema de controle em lidar com incertezas. Entende-se por incertezas, não somente aquelas provenientes da modelagem da planta a ser controlada, mas também os eventuais distúrbios sofridos por ela em operação. A relação abaixo, complementa e detalha um pouco mais as principais fontes de incertezas em um sistema dinâmico:

- Erros nos valores dos parâmetros utilizados na equações provenientes da modelagem (massa, momento de inércia, coeficiente de atrito, etc);
- Erros associados aos instrumentos de medição, no caso em particular, provenientes do próprio MSFS;
- Erros provenientes da linearização das equações diferenciais do modelo. Trata-se das simplificações realizada nas equações para torná-las lineares. No caso em questão, seria consideração de que os ângulos de atitude a serem corrigidos quando o helicóptero está em *hovering* são muito pequenos;
- Os distúrbios sofridos pela planta de uma forma geral.

Dentre os diversos controladores já desenvolvidos para o controle de sistemas dinâmicos, existe uma família que tem como principal característica a robustez. São os chamados controladores robustos. As referências sobre este assunto utilizadas nesta dissertação são os trabalhos de Zhou e Doyle (1998), Dorf e Bishop(2009) e Sampaio et al. (2013c). Os dois primeiros contém a teoria geral sobre controle moderno e robusto, e o terceiro o desenvolvimento do controlador H_∞ do quadricóptero ASCTEC PELICAN, que será apresentado abaixo.

O projeto de um controlador para um sistema dinâmico consiste basicamente em realizar a síntese do controlador, ou seja, desenvolver o controlador propriamente, e testá-lo em condições sujeitas à incertezas. Caso o desempenho não seja satisfatório, deve-se alterar algum parâmetro do controlador sintetizado até que o resultado esperado seja obtido.

O sistema de controle do quadricóptero é um sistema que possui 6 entradas e 4 saídas. A entrada é um vetor de estados X constituído pelo ângulos de atitude (θ, ψ, ϕ) e a posição (x, y, z) do helicóptero no espaço. A saída é um vetor u com os sinais de controle para o incremento de velocidade dos 4 motores.

Considerando que a planta seja estabilizável e detectável, o que portanto nos permite realizar a síntese do controlador H∞, tem-se na figura abaixo o diagrama de blocos da planta aumentada.

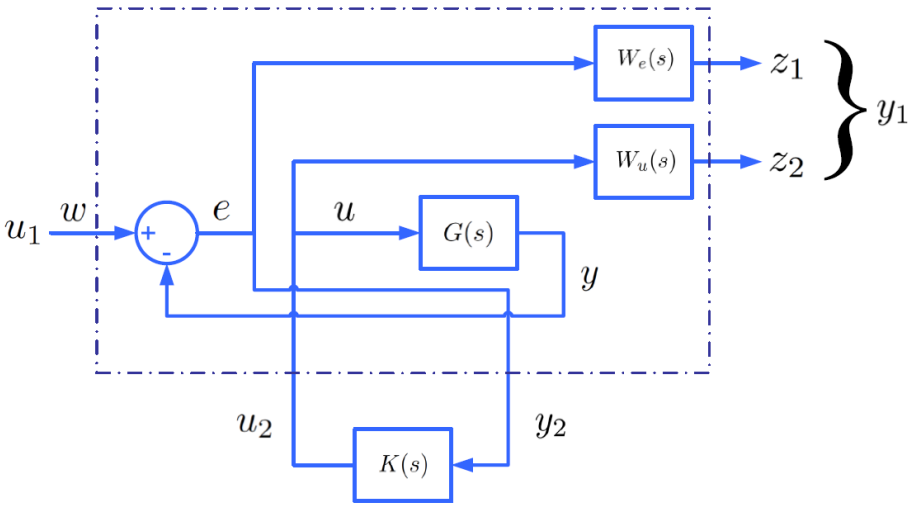


Figura 8 - Planta genérica aumentada G_{ap} do sistema a ser controlado pelo controlador H∞, representando a função de transferência T_{zw}.
Fonte: Sampaio et al. (2013c).

Com o auxílio do *Matlab* e de algoritmos iterativos baseados na abordagem $\gamma - iteration$, o valor de γ foi reduzido até seu ponto ótimo γ_{opt}. Com isso, obtém-se as funções de transferência das funções de ponderação do erro W_e, da ação de controle W_u, e do controlador $K(s)$, de maneira que a norma da função de transferência de malha fechada do sistema T_{zw} entre ω e $z_{1,2}$ satisfaça a condição:

$$\|T_{z\omega}\|_\infty = \left\| \begin{matrix} W_e S \\ W_u K S \end{matrix} \right\|_\infty = \left\| \begin{matrix} W_e S \\ W_u R \end{matrix} \right\|_\infty < \gamma \tag{4}$$

Onde S é uma função de sensibilidade e R a função transferência entre a ação de controle e a entrada de referência. Para se garantir estabilidade e robustez em relação aos distúrbios, incertezas e ruídos, é necessário que os ganhos de S sejam baixos em baixas frequências, ou seja, $|W_e S| \leq 1$. Por outro lado, os ganhos de R devem ser baixos

em altas frequências para se garantir o mesmo nível de rejeição aos distúrbios, ou seja, $|W_u R| \leq 1$. Esses dois critérios de robustez estão diretamente relacionados a:

- Falta de precisão nos parâmetros do modelo;
- Redução a zero do erro estacionário;
- Robustez às incertezas e variações em malha aperta;
- Robustez à ruídos inseridos na planta.

A aplicação do algoritmo $\gamma - iteration$ resultou num $\gamma_{opt} = 0.1915$, totalmente em acordo com a condição

$$\gamma > \gamma_{opt} = \underset{min}{\overset{K\ Stability}{}} \|T_{z\omega}\|_\infty \tag{5}$$

Para $\gamma < 1$. As funções de transferência de ponderação do erro W_e e da ação de controle W_u encontradas ao término do procedimento foram

$$W_e(s) = \frac{0.1s + 30}{s + 0.03} \tag{6}$$

$$W_u(s) = \frac{s + 0.06667}{330s + 10} \tag{7}$$

A Figura 9 mostra a resposta em frequência da função S que, como era de se esperar, possui ganhos menores em baixas frequências.

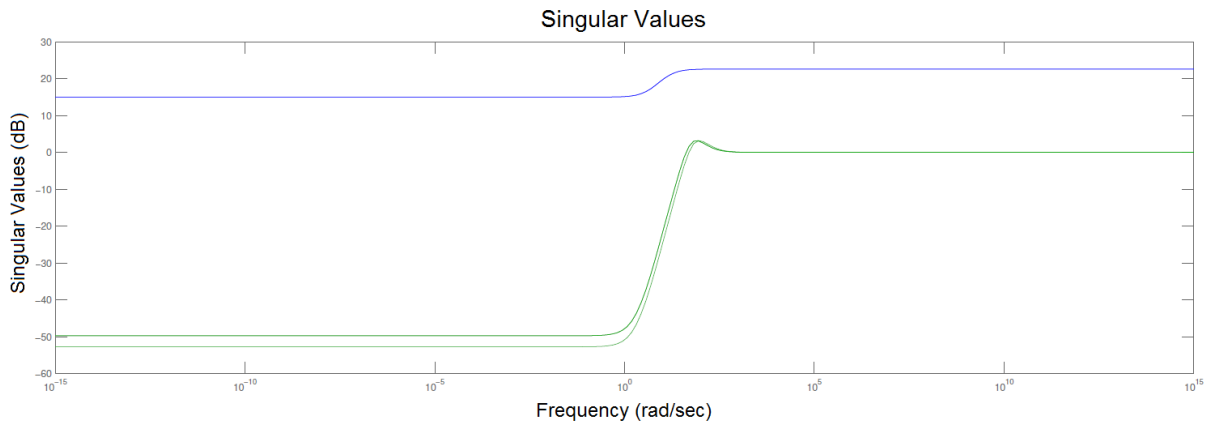


Figura 9 - Resposta em frequência da função S. Ganhos menores em baixas frequências.
Fonte: Sampaio et al. (2013c).

A Figura 10 mostra a resposta em frequência da função R que, como era de se esperar, possui ganhos menores em altas frequências.

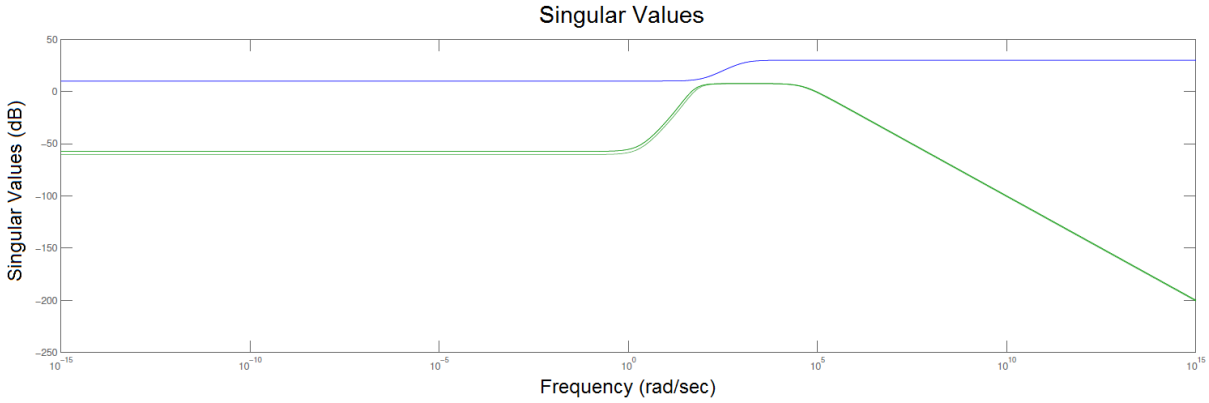


Figura 10 - Resposta em frequência da função R. Ganhos menores em altas frequências.
Fonte: Sampaio et al. (2013c).

A função de sensibilidade S ainda deve, idealmente, satisfazer o pico de sensitividade M_s e também a largura de banda de malha fechada ω_b, respeitando a seguinte relação:

$$|S(s)| \leq \left| \frac{s}{\frac{s}{M_s} + \omega_b} \right| \tag{8}$$

O melhor valor para ω_b é o valor da frequência natural não amortecida do sistema, ou seja $\omega_b \approx \omega_n$. Esses 2 parâmetros devem ser retirados dos gráficos de resposta em frequência. Para este caso, $M_s = 105$ e $\omega_b = 40\ rad/s$.

Ao final do procedimento de síntese, obtém-se o controlador K através das equações (6) e (7). O controlador K é constituído pelas 4 matrizes dinâmicas utilizadas no controle por variáveis de estado: $A_k, B_k, C_k\ e\ D_k$.

O Algoritmo 1 representa a lei de controle utilizada na implementação do controle H∞ nessa dissertação, tanto nas arquiteturas SIL quando HIL. Note que temos 2 vetores de entrada X e X_d, sendo um com os estados atuais do sistema e o outro com os estados desejados, e um vetor de saída, u, com os sinais de controle a serem atuados nos motores. Especificamente para a situação em que o helicóptero está em *hovering*, o X_d terá sempre valores nulos.

Algoritmo 1 - Lei de controle.

Entradas:	Vetor de estados $\boldsymbol{X} = \{\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$
	Vetor de estados desejados $X_d = \{\theta_d, \phi_d, \psi_d, x_d, y_d, z_d\}$
Saídas:	Vetor de sinais de controle para atuação nos motores $u = \{\Omega_1, \Omega_2, \Omega_3, \Omega_4\}$

Begin
  For *All State Variables* **do**
    [X] = *Read(States)*
  End
  $e \leftarrow [X_d - X]$
  $x_k \leftarrow A_k \cdot x_k + B_k \cdot e$
  $u \leftarrow C_k \cdot x_k + D_k \cdot e$
End

Fonte: Adaptado de Sampaio et al. (2013c).

A seção 4.1.11 detalha mais esse algoritmo.

4. Desenvolvimento

4.1. Desenvolvimento do Hardware

O *hardware* utilizado neste trabalho foi totalmente desenvolvido pelo autor entre os anos 2011 e 2012. O software de editoração eletrônica utilizado foi o *Altium Designer*®, e a programação feita em linguagem C no software MPLAB disponibilizado pela própria Microchip, fabricante do microcontrolador. Na ocasião de seu desenvolvimento, este *hardware* era destinado ao mercado de manutenção automotiva e tinha como objetivo calcular a potência de motores de combustão interna através da medição de aceleração e velocidade do veículo. Isso justifica a presença de alguns circuitos não utilizados neste trabalho, como por exemplo o *display* gráfico, teclado numérico, cartão de memória *flash* do tipo *SD Card*, *real-time clock*, acelerômetro e o circuito de medição de rotação por pinça indutiva. Além desses, ainda existem os circuitos de fonte de alimentação, microcontrolador e conversor USB que, por serem utilizados neste projeto, serão detalhados nos próximos tópicos.

4.1.1. Microcontrolador

Para este trabalho, o microcontrolador escolhido foi o Microchip dsPIC33FJ128MC706A. O dsPIC33FJ128MC706A é um microcontrolador de 16 bits do tipo DSC (*Digital Signal Controller*) que possui, além dos recursos comuns encontrados em microcontroladores, outros de DSP (*Digital Signal Processor*). A Tabela 7 mostra as características desse microcontrolador relevantes para esta aplicação (dsPIC33FJ128MC706A Datasheet) e (I2C dsPic33).

Tabela 7 - Características técnicas relevantes do dsPIC33FJ128MC706A

Característica	Valor	Unidade	Observações
Velocidade	40	MIPS	Tempo de instrução: 25nS
Consumo de corrente	70	mA	Este valor é diretamente proporcional à velocidade do processador.
SCI data rate	38 - 10M	bps	Velocidade da SCI
SPI data rate	15	MHz	Velocidade máxima da SPI
I2C data rate	100-1000	kHz	Velocidade máxima da I2C

4.1.2. Fonte de Alimentação

A placa precisa de 5,0V e 3,3V para alimentação de seus componentes. Também é desejável que ela funcione com baterias recarregáveis ou pilhas alcalinas. Para tanto, o circuito desenvolvido utilizou dois reguladores de tensão do tipo *charge pump buck-boost* da Microchip. O modelo escolhido foi o MCP1252-33X50.

O termo *"buck"* indica que o regulador consegue regular a tensão desejada, tendo como entrada uma tensão maior do que a de saída. O termo *"boost"* indica que ele consegue gerar a tensão de saída a partir de uma tensão de entrada menor do que a tensão de saída. Ou seja, o MCP1252-33X50 consegue regular a tensão de saída tendo como entrada valores menores ou maiores do que o valor regulado.

Este tipo de regulador é extremamente útil em aplicações que utilizam bateria (pilhas) como fonte geradora de tensão. O fato do regulador conseguir elevar a tensão, resulta num melhor aproveitamento das pilhas.

O MCP1252-33X50 pode regular 5,0V ou 3,3V em sua saída. Como as duas tensões são necessárias, foi utilizado 2 CIs na placa. A Figura 11 mostra o circuito utilizado na placa. A escolha do valor da tensão de saída é feita através da polarização (3,3V) ou aterramento (5,0V) do pino SELECT.

Dois outros recursos interessantes desse regulador são os pinos SHDN e PGOOD. O pino SHDN funciona como um pino de liga-desliga do regulador e o PGOOD como um supervisório do nível de tensão. Quando SHDN está aterrado (GND) a bomba está desligada. Este recurso é especialmente importante quando se deseja desligar parte do circuito com o intuito reduzir a corrente do circuito em momentos onde a parte desligada não se faz necessária. Neste trabalho, esse recurso não precisou ser utilizado.

O pino PGOOD deve ser ligado no microcontrolador com uma polarização de 10kΩ. Quando o regulador não conseguir manter a tensão desejada na saída (tolerância de 5%), este pino é aterrado. Note que, como necessitamos das duas tensões, não faz sentido monitorar o regulador de 3,3V, uma vez que a bomba de 5,0V já indicaria a falha bem antes dela ocorrer na bomba de 3,3V.

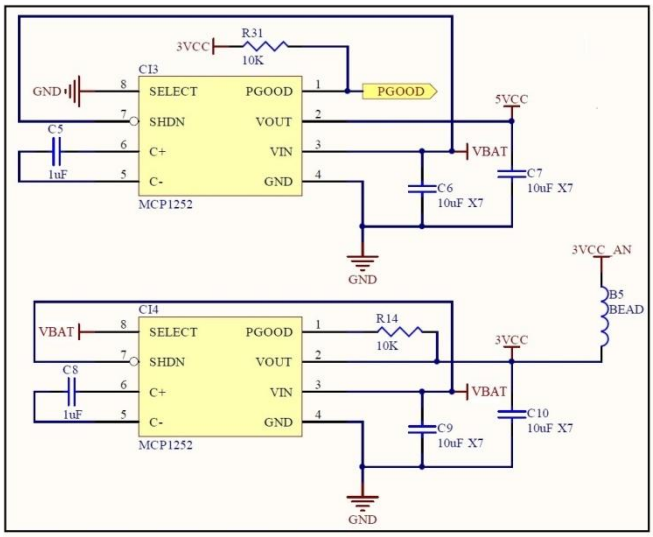


Figura 11 - Fonte de alimentação 5Vcc e 3,3Vcc utilizando o componente MCP1252-33X50

O valor de tensão máxima aceitável na entrada do MCP1252 é de 6,5V. O valor mínimo, embora o *datasheet* indique partir de 0,9V, depende da corrente total consumida pelo circuito. Considerando todos os componentes desta placa, verificou-se que o valor da corrente média de consumo não passou de 100mA. Com esse consumo, verificou-se, empiricamente, que a tensão mínima das baterias não poderia ser menor do que 3,0V.

Um outro requisito desejável para esta placa é a possibilidade de também alimentá-la com fonte externa de 12V. No entanto, esses dois reguladores não suportam mais do que 6,5V em suas entradas. Para tanto, desenvolveu-se também uma segunda fonte de alimentação de 5.0V com um regulador do tipo *buck* um pouco mais robusto. O modelo escolhido foi o LM2575-5.0, que regula em 5,0V tensões de entrada entre 7,5 e 60V. A Figura 12 ilustra o circuito desenvolvido, que deve ser visto como complemento do circuito da Figura 11.

Quando o *plug* P4 com a alimentação externa é conectado no circuito, os terminais 2 e 3 se separam, fazendo com que a referência (GNDPIL) proveniente das pilhas fique flutuando. Nesta situação, o terra do *plug* P4 passa a ser o terra do sistema que, depois de passar pelo CHOKE1, é dividido em outros 3 terras. Os PTCs 1 e 3 funcionam como fusíveis rearmáveis para proteção do circuito. Os diodos D1 e D2 estão presentes para proteger o circuito contra ligação invertida. O varistor de 40V funciona como um filtro e evita picos curtos de tensão (acima de 40V).

Na saída OUT do regulador tem-se a tensão de saída regulada (VBAT) e o circuito de chaveamento usado pelo regulador para garanti-la. O pino ON/OFF é uma chave de liga-desliga, semelhante ao pino SHDN da Figura 11. Quando ele é aterrado, o regulador está ligado e quando está polarizado com a tensão de saída, está desligado. Note que o LM2575 só está ligado quando houver alimentação externa da placa. Por fim, o transistor Q2 é apenas uma interface elétrica ligada no dsPIC para informá-lo de que a fonte externa está presente. Quando não existe *plug* P4 no *jack*, o GND do pino 3 fica em curto com o GNDPIL, desligando o LM2575 e colocando a tensão das pilhas no VBAT. Esse VBAT é a entrada de tensão do circuito da Figura 11.

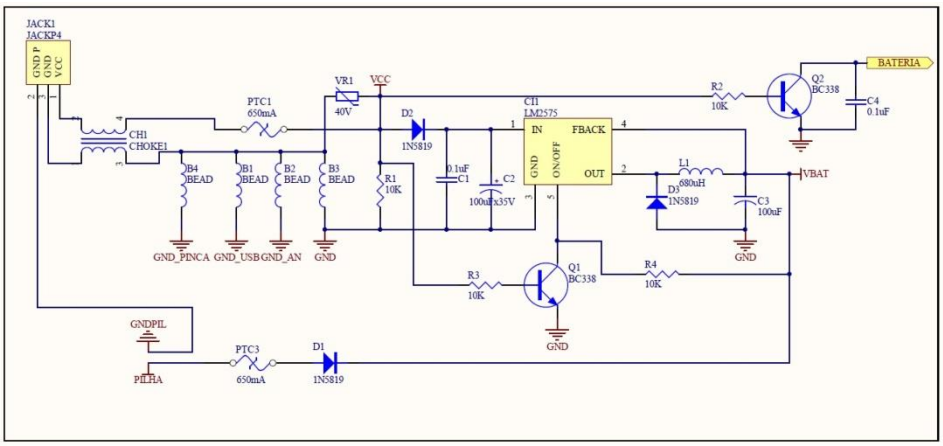


Figura 12 - Circuito de alimentação com LM2575-5.0

4.1.3. Conversor USB-SCI

Para se comunicar com o computador, utiliza-se um conversor serial (SCI) para USB, pois o dsPIC não possui comunicação USB internamente. O conversor utilizado foi o FT232BL do fabricante FTDI Chip, (FT232BL Datasheet). No lado USB do conversor, a velocidade de comunicação é de 480 Mbps (USB 2.0). No lado SCI, a velocidade de comunicação varia de 183 bps até 3 Mbps. A alimentação do CI deve ser entre 4,35 V e 5,25 V, porém possui recurso elétrico para comunicação de dados em 3,3 V, bastando para isso, ligar um de seus pinos em 3,3 V. Se for o caso, o CI ainda consegue alimentar o resto do circuito utilizando a alimentação proveniente do conector USB.

A Figura 13 ilustra as ligações básicas do FT232BL nesta aplicação. No conector USB do tipo B foram colocadas 4 ferritas para melhoria na imunidade a ruídos elétricos do circuito. A alimentação do CI é feita com a alimentação (VBUS) proveniente da

própria porta USB. Desta forma, quando a USB não está sendo utilizada, o CI permanece desligado e, consequentemente, sem consumir corrente do circuito. O cristal utilizado é de 6 MHz e os pontos USB_RX e USB_TX correspondem aos pinos de comunicação com o controlador. O pino VCCIO deve ser ligado em 3.3 V (3 VCC) para que a comunicação com o dsPIC fosse realizada neste nível de tensão.

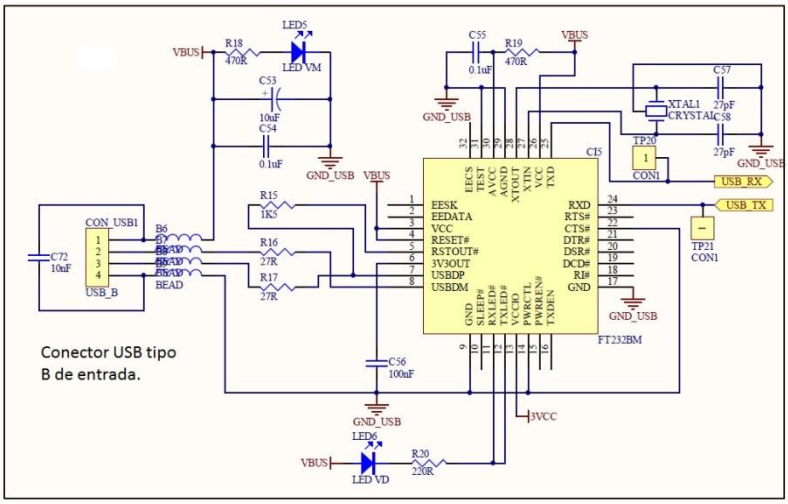


Figura 13- Circuito conversor Serial-USB FT232BL

Do ponto de vista do software, para que o dsPIC se comunique com o computador, utiliza-se um *driver* disponível pela fabricante do *chip*, FTDI chip, que transforma a porta USB numa porta COM virtual. O *download* desse *driver* pode ser feito diretamente no site da empresa.

4.1.4. *Flight Variable Management System* (FVMS V2.0)

Para interagir com o MSFS e com a placa externa é necessária a existência de um terceiro *software*. Inicialmente o *software* FVMS (*Flight Variables Management System*) foi desenvolvido em linguagem C/C# pelo ART, conforme descrito por Sampaio et al. (2013a). Para este trabalho, uma nova versão foi gerada em linguagem de programação Object Pascal (ferramenta DELPHI® 7.0). A escolha pela linguagem se deu pelos motivos citados abaixo:

- Linguagem de fácil aprendizado e com vasto material de suporte. Além da IDE de fácil utilização;
- Facilidade de acesso às API´s do Windows, o que proporciona mais flexibilidade de programação e maiores velocidades de execução;

- Disponibilidade de procedimentos de comunicação serial utilizando funções da API do Windows, ou seja, sem a utilização de bibliotecas prontas do DELPHI® ou de terceiros;
- Orientação a eventos do DELPHI® 7.0, que facilita a execução de eventos com o uso de timers;
- Interface gráfica com componentes gráficos de fácil utilização;

A comunicação com o MSFS é feita através de uma *dynamic link library* (DLL) desenvolvida por Peter Dowson (DOWSON, 2014), chamada FSUIPC.DLL. Essa DLL permite a leitura e gravação de aproximadamente 2000 variáveis de voo. O que o FVMS faz, portanto, é a leitura de algumas dessas variáveis (sensores) e, posteriormente, a gravação de outras (atuadores). Entre a leitura e a gravação das variáveis de voo, ocorre a execução do controle. No caso do SIL ele é feito no próprio FVMS e no do HIL é feito no *hardware* externo. A Figura 14 ilustra a relação entre o FVMS, o MSFS e o *hardware* externo.

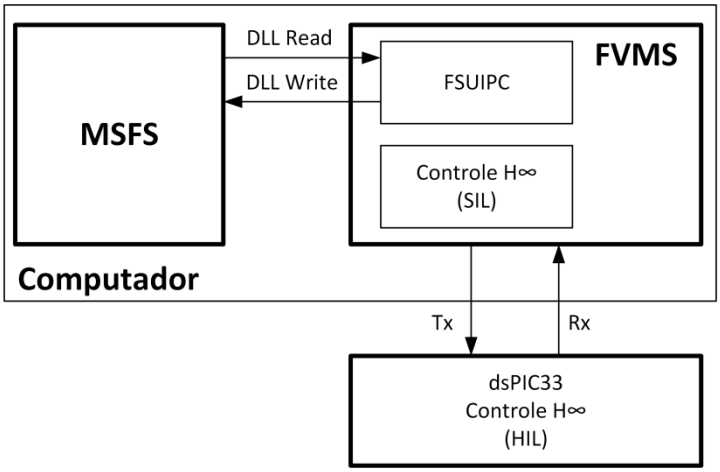


Figura 14 - Relação entre MSFS, FVMS e FSUIPC

4.1.5. Detalhes da comunicação com FSUIPC

Para que a comunicação entre o FVMS e o MSFS possa ocorrer através da DLL FSUIPC, é necessário executar o procedimento de inicialização. O MSFS já deve estar rodando para que o procedimento funcione corretamente.

A DLL FSUIPC possui basicamente 5 funções: *FSUIPC_Open()*, *FSUIPC_Close()*, *FSUIPC_Read()*, *FSUIPC_Write()* e *FSUIPC_Process()*. As tabelas a seguir descrevem o significado de cada uma delas, explicitando os parâmetros de entrada e de saída.

Na prática, o procedimento *FSUIPC_Open* deve ser realizado na inicialização do programa, o *FSUIPC_Close* na finalização e os procedimentos *FSUIPC_Read* e *FSUIPC_Write* quando se deseja ler ou gravar alguma informação nas posições de memória. O procedimento *FSUIPC_Process* sempre deve ser executado posteriormente à esses dois últimos procedimentos citados.

Tabela 8 - Função FSUIPC_Open, executada na inicialização do programa

function FSUIPC_Open(dwFSReq: Cardinal, **var** dwResult: Cardinal): **Boolean**;	
Função:	Inicia a comunicação com o MSFS
Resultado da função:	True: conseguiu realizar a comunicação com o MSFS False: não conseguiu realizar a comunicação com o MSFS
Variáveis de entrada:	dwFSReq e dwResult
Variáveis de saída:	dwResult O dwResult retornará o resultado ocorrido na execução da função.

Tabela 9 - Função FSUIPC_Close, executada na finalização do programa

Procedure FSUIPC_Close;	
Função:	Finaliza a comunicação com o MSFS
Resultado da função:	---
Variáveis de entrada:	---
Variáveis de saída:	---

Tabela 10 - Função FSUIPC_Read, executada na leitura dos dados do MSFS

function FSUIPC_Read(dwOffset : DWORD; dwSize : DWORD; pDest : Pointer; **var** dwResult : DWORD) : **Boolean**;	
Função:	Executa a leitura de dwSize bytes da posição de memória indicada em dwOffset
Resultado da função:	True: leitura realizada False: leitura não realizada
Variáveis de entrada:	dwOffset: endereço inicial a ser lido na memória do computador dwSize: quantidade de bytes que devem ser lidos pDest: indica o ponteiro inicial do array no qual os dados serão colocados
Variáveis de saída:	dwResult O dwResult retornará o resultado ocorrido na execução da função.

Tabela 11 - Função FSUIPC_Write, executada na escrita dos dados no MSFS

function FSUIPC_Write(dwOffset : DWORD; dwSize : DWORD; pSrce : Pointer; var dwResult : DWORD) : Boolean;	
Função:	Executa a gravação de dwSize bytes na posição de memória indicada em dwOffset
Resultado da função:	True: gravação realizada False: gravação não realizada
Variáveis de entrada:	dwOffset: endereço inicial a ser lido na memória do computador dwSize: quantidade de bytes que devem ser lidos pSrce: indica o ponteiro inicial do array no qual os dados serão colocados
Variáveis de saída:	dwResult O dwResult retornará o resultado ocorrido na execução da função.

Tabela 12 - FSUIPC_Process, executada sempre na sequência das funções FSUIPC_Read e FSUIPC_Write

function FSUIPC_Process(var dwResult : DWORD) : Boolean;	
Função:	Executado sempre na sequência de um FSUIPC_Write ou FSUIPC_Read para tratamento da mensagem de erro (dwResult) e carregamento dos bytes na posição indicada pelos ponteiros pSrce e pDest.
Resultado da função:	True: processamento realizado False: processamento não realizado
Variáveis de entrada:	-
Variáveis de saída:	dwResult O dwResult retornará o resultado ocorrido na execução da função.

4.1.6. Detalhes da comunicação com o dsPIC

A comunicação com o dsPIC é feita através da comunicação USB da placa. O dsPIC utilizado não possui comunicação USB, portanto, foi necessária a utilização de um CI de conversão SCI - USB. O CI escolhido foi o FT232BL, do fabricante FTDI Chip (FT232BL Datasheet). A velocidade de comunicação usada foi de 115200 bps. A utilização do chip de conversão implica na instalação de um *driver* no computador que transforma a porta USB numa porta serial virtual, ou seja, do ponto de vista da programação, a comunicação entre os dois dispositivos acontece como se o mesmo utilizasse uma porta serial RS232 comum.

Protocolo de comunicação

O protocolo de comunicação serial usado neste trabalho teve como premissa usar a menor quantidade possível de dados durante as transmissões. As informações que devem ser enviadas pelo FVMS para o *hardware* externo são: ângulos atuais (*roll, pitch*

yaw), posição atual (x, y, z), ângulos desejados (*roll, pitch, yaw*) e posições desejadas (x, y, z). Ou seja, são 12 variáveis do tipo *float-point* compostas por 4 bytes (32 bits) cada. Na recepção, o FVMS recebe apenas 4 variáveis também do tipo *float-point* com 4 bytes cada, correspondente ao incremento de velocidade de cada um dos 4 motores. A Figura 15 ilustra essa transferência de dados.

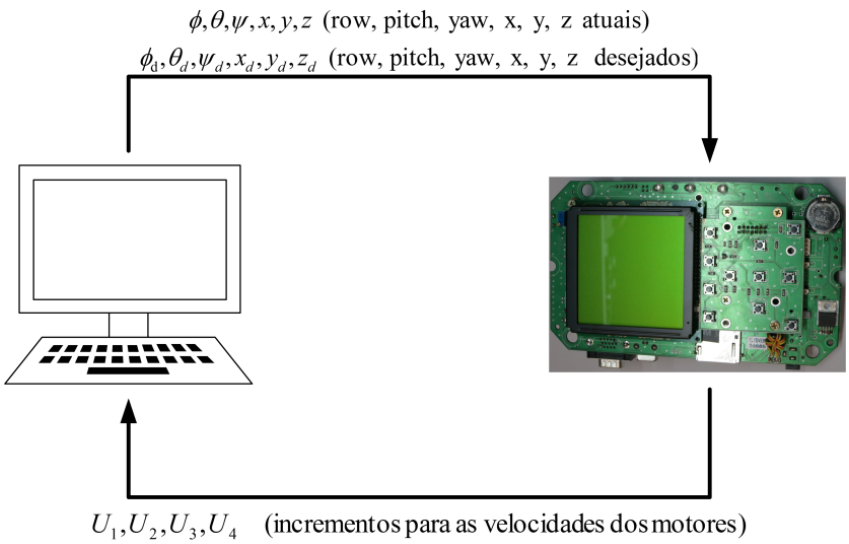


Figura 15 - Dados enviados e recebidos pelo FVMS

Além desses *bytes*, ainda há um para indicação da quantidade de *bytes* a serem enviados e outros 2 *bytes* de *checksum*. O *byte* de quantidade é o primeiro a ser enviado e os de *checksum* são os 2 últimos de cada *frame*. Esses 2 *bytes* de *checksum* correspondem aos *bytes* MSB e LSB da somatória simples de todos os *bytes* enviados anteriormente. Contudo, o *frame* de envio é constituído por 51 *bytes* e que o de resposta por 19, conforme ilustra a Tabela 13.

Tabela 13 - *Frames* de envio e recepção serial

Frame de envio (TX)			
Quantidade	Ângulos	*Checksum*	Total
1	12*4 = 48	2	51
Frame de recepção (RX)			
Quantidade	Ângulos	*Checksum*	Total
1	4*4 = 16	2	19

4.1.7. Variáveis lidas do MSFS e variáveis escritas no MSFS

Antes de executar o controle, é necessária a leitura das variáveis pertinentes à atitude da aeronave. A Tabela 14 mostra quais são essas variáveis e seus respectivos

endereços no espaço reservados para memória do MSFS. Dentre elas estão as variáveis utilizadas no controle a algumas outras utilizadas apenas pela interface gráfica do FVMS.

Da mesma forma, a Tabela 15, mostra as variáveis que devem ser gravadas depois que o controle for calculado. Note que não é possível o acesso aos motores do quadricóptero por uma limitação do MSFS. O MSFS permite apenas a edição das 3 superfícies de controle de posição normalmente utilizadas em aviação: *elevator*, *aileron* e o *rudder*.

Tabela 14 - Variáveis lidas do Flight Simulator

Símbolo	Descrição	Endereço na memória
ϕ	Ângulo de rolagem (Roll)	057C
θ	Ângulo de arfagem (Pitch)	0578
ψ	Ângulo de guinada (Yaw)	0580
$\dot{\phi}$	Velocidade de rolagem	30B0
$\dot{\theta}$	Velocidade de arfagem	30A8
$\dot{\psi}$	Velocidade de guinada	30B8
$\ddot{\phi}$	Aceleração de rolagem	3080
$\ddot{\theta}$	Aceleração de arfagem	3078
$\ddot{\psi}$	Aceleração de guinada	3088
$\dot{x}$	Velocidade ao longo do eixo x	3198
$\dot{y}$	Velocidade ao longo do eixo y	31A0
$\dot{z}$	Velocidade ao longo do eixo z	3190
$\ddot{x}$	Aceleração ao longo do eixo x	31C0
$\ddot{y}$	Aceleração ao longo do eixo y	31C8
$\ddot{z}$	Aceleração ao longo do eixo z	31D0

Tabela 15 - Variáveis que devem ser gravadas no Flight Simulator

Descrição	Endereço na memória
Elevator (pitch)	0BB2
Aileron (roll)	0BB6
Rudder (Yaw)	0BBA

4.1.8. Interface gráfica do *software*

A interface do FVMS v2.0 pode ser dividida em 3 partes: Painel de instrumentos de visualização em tempo real, painel de gráficos gerados em tempo real e painel de funções, configurações e ajustes. A Figura 16 é o *print screen* da tela do programa.

O painel de instrumentos para visualização em tempo real, localizado à direita da interface mostra algumas informações lidas do FVMS em tempo real, tais como os ângulos de rolagem, arfagem e guinada, suas respectivas velocidades e acelerações,

altitude, posição das superfícies de controle *elevator*, *aileron* e *rudder*, e as velocidades e acelerações de translação nos eixos x, y e z. A Figura 17 mostra com mais detalhes esta parte da tela.

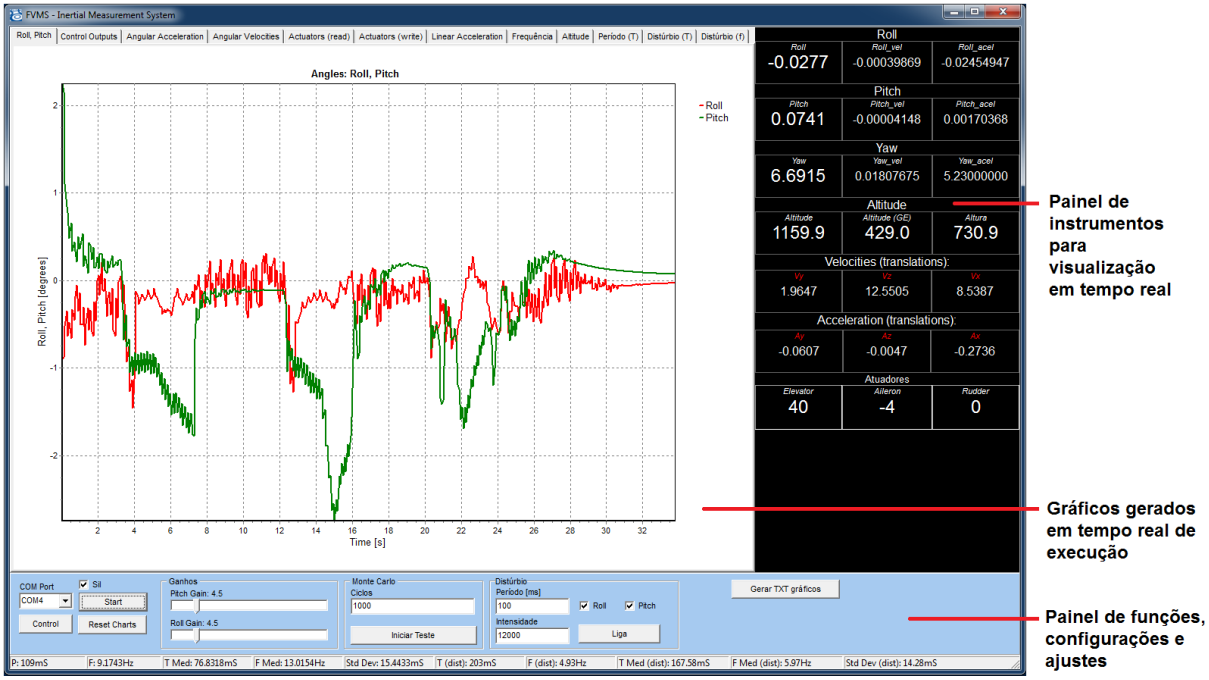


Figura 16 - Interface gráfica do FVMS v2.0.

Figura 17 - Painel de instrumentos para visualização em tempo real.

O painel de gráficos gerados em tempo real possui 12 abas com 12 gráficos gerados em tempo real. A maioria deles são os mesmos gráficos gerados durante os testes e apresentados no capítulo 5 deste trabalho. Outros armazenam informações que foram analisadas durante o desenvolvimento dos testes, porém não foram relevantes no escopo da dissertação. A Figura 18 mostra em detalhes apenas a aba de escolha dos gráficos.

Figura 18 - Detalhe da aba de escolha dos gráficos.

O painel de funções, configurações e ajustes, localizado na parte inferior da tela e em detalhe nas figuras Figura 19 e Figura 20, contém os ajustes dos ganhos para atuação nas superfícies *elevator* e *aileron*, a caixa de seleção para escolha da porta serial de comunicação com o *hardware* externo, os botões de liga e desliga da varredura do controle e a caixa de seleção da opção SIL. Quando a opção SIL está selecionada, a execução do controle acontece dentro do próprio FVMS (SIL). Quando a opção não está selecionada, o controle é executado no *hardware* externo (HIL). Ao pressionar o botão "*Start*", a varredura de controle se inicia. Sua finalização só ocorre quando o botão for pressionado novamente. Para a execução do controle, foi utilizado o componente TTimer do DELPHI® com a propriedade "*interval*" configurada para 1ms. Na verdade, essa configuração indica para o *software* que as execuções do controle devem ser realizadas sequencial e ininterruptamente.

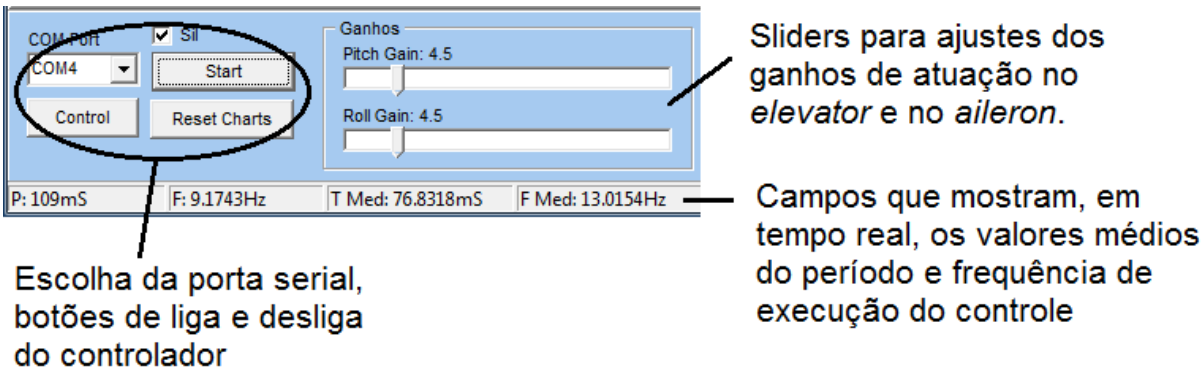


Figura 19 - Detalhe do painel de funções, configurações e ajustes (lado esquerdo).

A Figura 20, mostra duas funções específicas utilizadas nesse trabalho. A primeira delas é a função "Teste de Monte Carlo", usada para aferição da frequência de acesso às informações do MSFS. Veja mais detalhes sobre isso na seção 5.2. A segunda é a função "Distúrbio", que nada mais é do que uma função que adiciona um distúrbio periódico nas superfícies de controle durante o voo controlado do helicóptero. Veja mais detalhes sobre essa função nos itens 4.1.9 e 4.1.10. Nesta mesma figura, ainda pode ser visto o botão "Gera TXT gráficos", usado para exportar os gráficos mostrados nos resultados do capítulo 5 para edição no MATLAB®.

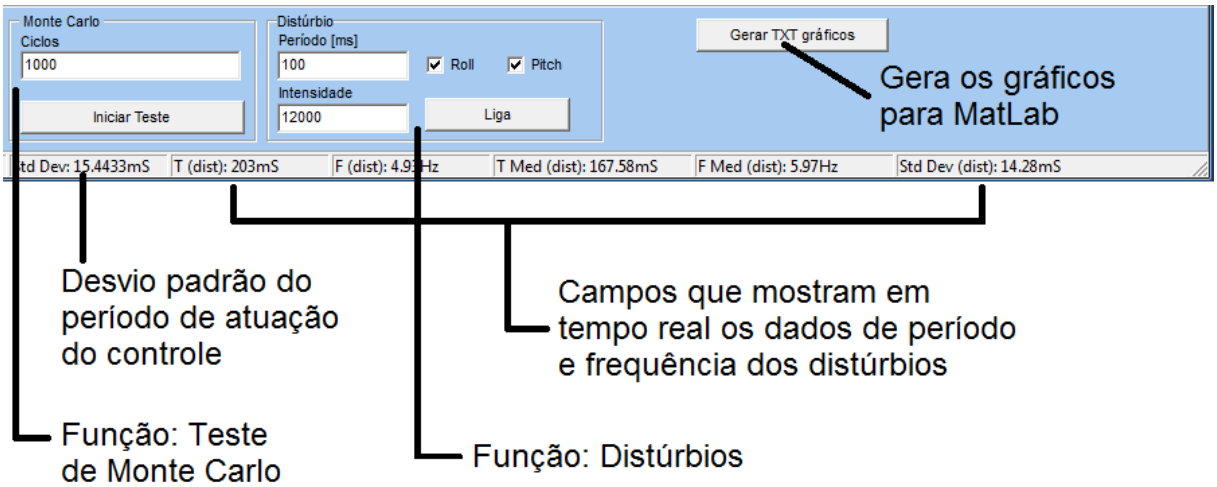


Figura 20 - Detalhe do painel de funções, configurações e ajustes (lado direito).

Na extremidade inferior da tela ainda tem alguns campos para visualização de mais algumas informações em tempo real. São informações referentes ao período, frequência e desvio padrão da execução do controle (Figura 19) e as mesmas, porém referentes aos distúrbios aplicados durante o voo (Figura 20). Veja mais informações sobre essas funções nos itens 4.1.9 e 4.1.10.

4.1.9. Distúrbios

Com o objetivo de avaliar melhor a atuação do controle H_∞ implementado quando o helicóptero está em condições desfavoráveis, foi implementada uma função que tem por objetivo aplicar um distúrbio controlado e periódico nos atuadores da aeronave. Para tanto, um componente timer foi utilizado. A Figura 20 mostra a interface de

entrada de dados para esta função. Nela, pode-se definir a intensidade do distúrbio, o local de aplicação e o período em que ele acontece.

O valor máximo que se consegue gravar nesses atuadores é 16384. O campo "intensidade", portanto, indica qual é o valor que deve ser gravado. O campo "período", corresponde ao valor utilizado na propriedade "*interval*" do componente TTimer utilizado no DELPHI® e isso significa que, a cada período de tempo (ms) a rotina de distúrbio é executada. Como o sistema operacional e o DELPHI® não são sistemas que trabalham em tempo real, infelizmente o valor desse período não pode ser garantido. Nos testes, portanto, ressaltaremos os valores de período e frequência resultante medidos com base no relógio do computador.

A Figura 21 mostra a relação do distúrbio e a execução do controle. Os 2 procedimentos são implementados com o componente TTimer do DELPHI®, que trabalham em duas *threads* separadas, ou seja, em paralelo. Note que os dois atuam nas mesmas variáveis.

As duas caixas de seleção "*Roll*" e "*Pitch*" servem simplesmente para informar onde o distúrbio deve ser aplicado. O botão "Liga" deve ser pressionado no momento em que se desejar que o distúrbio se inicie. Ao pressionar novamente, a função é desligada.

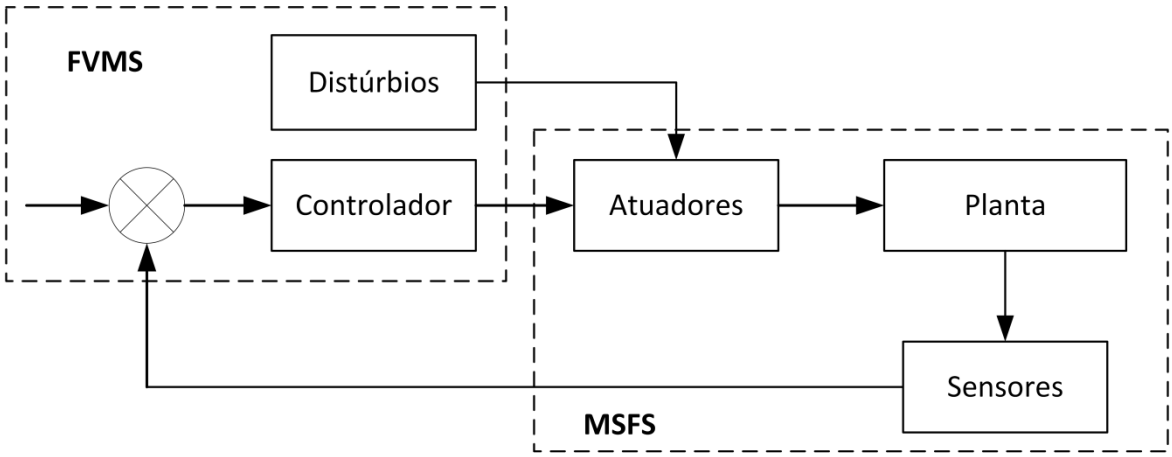


Figura 21 - Distúrbio sendo aplicado nos atuadores.

4.1.10. Cálculos do período, frequência e desvio padrão

Conforme mencionado no item 4.1.8 e 4.1.9, tanto a rotina de controle quanto a de distúrbios foram implementada usando o componente TTimer do DELPHI®. No

entanto, conforme mencionado anteriormente, como o sistema operacional e o DELPHI® não funcionam em tempo real, os períodos configurados nos componentes não se realizam na prática. Para que os resultados gerados fossem os mais realísticos possíveis, foram implementados procedimentos para aferição desses tempos.

A ideia básica foi pegar o tempo de *clock* do computador a cada execução do evento "*OnTimer*" e fazer a diferença com o valor da execução anterior. Esse valor corresponde ao período atual de execução das rotinas. A frequência atual corresponde simplesmente ao inverso do valor do período. Os gráficos de período e frequência apresentados no capítulo 5 referem-se à esses valores. Nas abscissas está sempre o tempo corrido do teste e nas ordenadas o valor do período ou frequência quando a interrupção do *timer* aconteceu.

Durante os testes constatou-se, no entanto, que o valor do período não era constante. Daí a necessidade do cálculo da média desses valores, tanto do período quanto da frequência. Para validação da acurácia desses valores, utilizou-se o desvio padrão. A segunda curva dos gráficos de período e frequência apresentados no capítulo 5, corresponde a esses valores médios.

A quantidade de amostras utilizadas no cálculo da média aritmética e do desvio padrão na rotina de controle foi de até 500, ou seja, os dados do período de cada amostra foram colocados numa estrutura de dados do tipo fila. Durante as primeiras 500 amostras, a quantidade utilizada nas contas era a quantidade de dados recebidos até o momento. Ao passar das 500 amostras, a fila começou a rodar e armazenar sempre as últimas 500 amostras.

No caso das médias nas rotinas de distúrbio, a quantidade total de amostras foi de até 50 itens apenas. Em geral, o tempo em que o distúrbio fica ligado é restrito à alguns segundos, portanto uma quantidade muito grande de amostras nunca seria atingida.

As fórmulas utilizadas nos cálculos da média aritmética e do desvio padrão estão descritas no capítulo 5, mais especificamente nas equações (11), (12) e (13).

4.1.11. Algorítmo de controle executado

Algoritmo 2 corresponde ao controle executado tanto no SIL quanto no HIL. No caso do SIL, ele era executado pelo próprio FVMS e no HIL, pelo dsPIC. As entradas do procedimento são dois vetores, um com os estados atuais e outro com os valores desejados. As variáveis correspondem aos ângulos de atitude e ao posicionamento do quadricóptero: *roll* (ϕ), *pitch* (θ), *yaw* (ψ), x, y e z. Consulte as figuras Figura 5 e Figura 6 para melhor visualização dessas grandezas. As saídas do procedimento são os incrementos de velocidade dos 4 motores: u_1, u_2, u_3, u_4.

Algoritmo 2 - Código de execução do controle H∞

```
Input:  state = [φ, θ, ψ, x, y, z]^T
        desired = [φ_d, θ_d, ψ_d, x_d, y_d, z_d]^T
Saídas: Sinais de controle individuais de cada motor u_1, u_2, u_3, u_4
wHILe TRUE do
        if serial.hasDATA then begin
                for i=1 to 6 do begin
                        erro[i] = desired[i] − state[i]
                end
                for i=1 to 6 do begin
                        for j=1 to 6 do begin
                                x_k[i] = A_k[i][j].state[j] + B_k[i][j].error[j]
                        end
                end
                for i=1 to 4 do begin
                        for j=1 to 6 do begin
                                u[i] = C_k[i][j].x_k[j]
                        end
                end
        end
end
```

Infelizmente o MSFS não entende a existência dos 4 motores do quadricóptero. Quando uma aeronave possui hélice voltada para cima, independente da quantidade, para o MSFS essas hélices são responsáveis apenas pela aceleração da aeronave, como um helicóptero comum. Os atuadores disponibilizados pelo MSFS, independente do tipo de aeronave, são os atuadores utilizados no controle de atitude de um avião comum: *elevator (pitch), aileron (roll)* e *ruder (yaw)*. Portanto, os dados de saída do controlador não podem ser enviados diretamente para o MSFS sem algum mapeamento que transforme velocidade dos 4 motores em atuação angular.

Como o objetivo deste trabalho é controlar somente os ângulos de *roll* e *pitch* do quadricóptero, isso facilita consideravelmente esse mapeamento. Conforme indicado na Figura 5, a arfagem (*pitch*) e a rolagem (*roll*) ocorrem quando o quadricóptero faz rotação em torno dos eixos y e x, respectivamente. O giro em torno do eixo y é controlado pelos motores 1 e 3, enquanto que o giro em x é controlado pelos motores 2 e 4. Sendo assim, os valores recebidos em : u_1 e u_2 foram multiplicados por um ganho de ajuste, somados aos valores atuais e configurados nas variáveis *elevator* e *aileron* no MSFS, conforme indicado nas (9) e (10). O ganho foi verificado de forma empírica no FVMS v2.0 (Figura 19) avaliando-se a eficiência da estabilização no próprio MSFS.

$$Elevator = Elevator + u_1 * Ganho \tag{9}$$

$$Aileron = Aileron + u_2 * Ganho \tag{10}$$

4.1.12. *Software in the loop* (SIL)

No *Software in the loop* (SIL), o FVMS realiza a leitura das variáveis de voo do quadricóptero, executa o controle H∞ e, posteriormente, realiza a atuação nas superfícies de controle do avião no MSFS. A Figura 22 ilustra a arquitetura geral de funcionamento do SIL.

O FVMS lê as variáveis de interesse, executa o controle e grava as atualizações dos atuadores. Tanto a leitura quanto a gravação são executados com a DLL FSUIPC. Esta biblioteca acessa as posições de memória das variáveis de interesse utilizadas pelo MSFS.

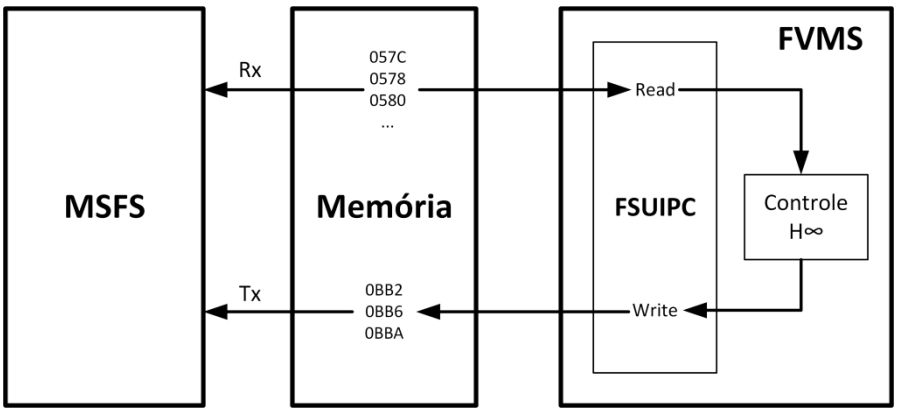


Figura 22 - Arquitetura *software in the loop* (SIL)

4.1.13. *Hardware in the loop* (HIL)

No *Hardware in the loop (HIL)*, o FVMS realiza a leitura das variáveis de voo do quadricóptero no MSFS, envia os ângulos via serial para a placa com o dsPIC, recebe pela mesma via como resposta o incremento de velocidade dos motores do quadricóptero e, posteriormente, realiza a atuação nas superfícies de controle do avião no MSFS. A Figura 23 ilustra a arquitetura geral de funcionamento do HIL.

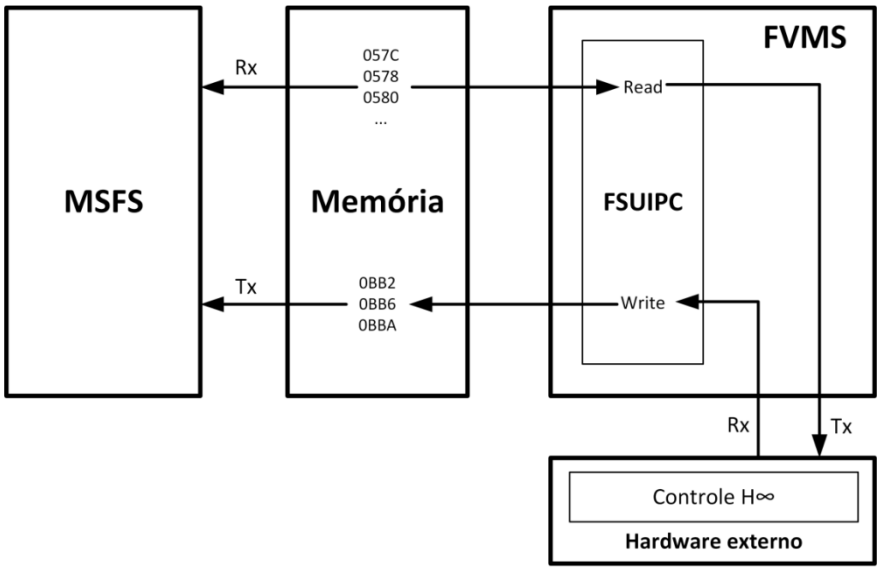


Figura 23 - Arquitetura *hardware in the loop* (HIL)

Note que, assim como no SIL, a DLL FSUIPC está na interface entre o MSFS e o FVMS. Ela é responsável por acessar os endereços das variáveis de interesse na memória do computador.

5. Testes e Resultados

Nesta seção são apresentados os resultados obtidos nesta pesquisa. O primeiro deles, descrido na seção 5.1, é o *hardware* desenvolvido. No item 5.2, é realizado o teste de Monte Carlo para validação dos tempos de acesso às informações no MSFS e dos tempos de comunicação. Da seção 5.3 em diante são apresentados alguns testes para validação da plataforma e do controlador H_∞ implementado. São basicamente 4 testes realizados:

- Decolagem e estabilização (item 5.3);
- Teste de estabilidade com distúrbio simples aplicado no joystick (item 5.4);
- Teste de estabilidade durante uma tempestade ligando e desligando o controle (item 5.5);
- Teste de estabilidade aplicando distúrbios sistemáticos forçados (item 5.6)

Cada teste foi realizado em 4 situações que se diferem em termos da arquitetura de controle e da condição de visualização do *software* simulador: SIL com a tela maximizada, SIL com a tela minimizada, HIL com a tela maximizada e HIL com a tela minimizada. Para cada uma dessas situações foram gerados ao menos 6 gráficos relevantes:

- Ângulos de atitude (*roll* e *pitch*);
- Sinais de controle;
- Atuação nas superfícies de controle;
- Frequência;
- Período;
- Altitude.

De uma forma geral, esses seis gráficos de cada teste mostram a variação dos ângulos de atitude e a respectiva atuação de controle agindo na correção desses ângulos. No entanto, como resultados mais importantes são destacadas as velocidades de atuação do controle em cada uma dessas condições descritas no parágrafo anterior. A ideia é avaliar a controlabilidade da aeronave diante às diferentes condições implantadas durante os testes.

Nos 2 primeiros testes (seções 5.3 e 5.4), o foco fica sobre a eficiência geral do controlador e as velocidades de execução do controle. No terceiro teste (5.5), a aeronave é colocada numa tempestade (configurada no MSFS) e o controle ligado somente quando ela estiver na iminência de perder a estabilidade. A intenção deste ensaio é avaliar a eficiência do controlador nesta situação apenas. No último teste (5.6), o helicóptero é exposto a distúrbios sistemáticos gerados nas superfícies de controle. Outros dois gráficos são gerados mostrando a frequência e o período de atuação do distúrbio no sistema e as avaliações são feitas com base no impacto gerado pelos distúrbios na estabilidade geral da aeronave.

A Tabela 16 mostra os resultados de frequência e período de atuação do controle obtidos nos dois primeiros testes (seções 5.3 e 5.4). É interessante observar a coerência desses resultados com os resultados a serem apresentados no teste de Monte Carlo (seção 5.2, Tabela 18).

A Tabela 17 contém a média aritmética considerando os 2 testes mencionados na Tabela 16. Essa tabela será comparada com a Tabela 20 na seção 5.6 onde se discute o impacto dos distúrbios sobre a frequência de controle.

Tabela 16 - Frequência e Período de execução do controle nos testes de decolagem e de estabilidade com distúrbios simples aplicados no joystick.

Teste	Arq.	Tela	f_{min} [Hz]	f_{max} [Hz]	f_{med} [Hz]	T_{min} [ms]	T_{max} [ms]	T_{med} [ms]
Decolagem	SIL	Maximizada	5,81	6,41	5,89	156	172	169,68
		Minimizada	12,82	21,28	14,35	47	78	69,71
	HIL	Maximizada	5,81	6,41	5,89	156	172	169,74
		Minimizada	9,17	16,13	11,03	62	109	90,67
Estabilização com distúrbios no joystick	SIL	Maximizada	5,81	6,41	5,89	156	172	169,65
		Minimizada	12,82	21,74	14,30	46	78	69,918
	HIL	Maximizada	5,81	6,41	5,89	156	172	169,67
		Minimizada	9,09	12,82	11,025	78	110	90,71

Tabela 17 - Resumo (Valores médios)

Arquitetura	Tela	f_{med} [Hz]	T_{med} [ms]
SIL	Maximizada	5,89	169,67
	Minimizada	14,33	69,81
HIL	Maximizada	5,89	169,72
	Minimizada	11,03	90,69

5.1. *Hardware* desenvolvido

O primeiro resultado obtido com esse trabalho foi a placa com o dsPIC. Com descrito anteriormente, essa placa possui diversos outros recursos não utilizados nesse trabalho e que foram executados para que ela pudesse ser utilizada em outros projetos do grupo ART. Os recursos mais importantes no contexto deste trabalho são: a escolha do microcontrolador, a fonte de alimentação necessária para seu funcionamento e o conversor USB.

A Figura 24 mostra o lado superior da placa. Nela podem ser localizados os seguintes recursos: teclado, *display*, conector DD15 para conexão com acelerômetro externo, conector de *SD Card*, circuito de *real time clock* e parte do circuito da fonte de alimentação (LM2575).

Figura 24 - Placa com dsPIC: Vista superior

A Figura 25 mostra a parte inferior da placa, onde estão localizados a maioria dos circuitos em SMD. Nela, podem ser visualizados os seguintes circuitos: o microcontrolador, o circuito da fonte de alimentação, conversor USB, o *beep* (sinalizador sonoro), circuito de *real time clock*, sensor barométrico, circuito do *display* gráfico e sensor de temperatura.

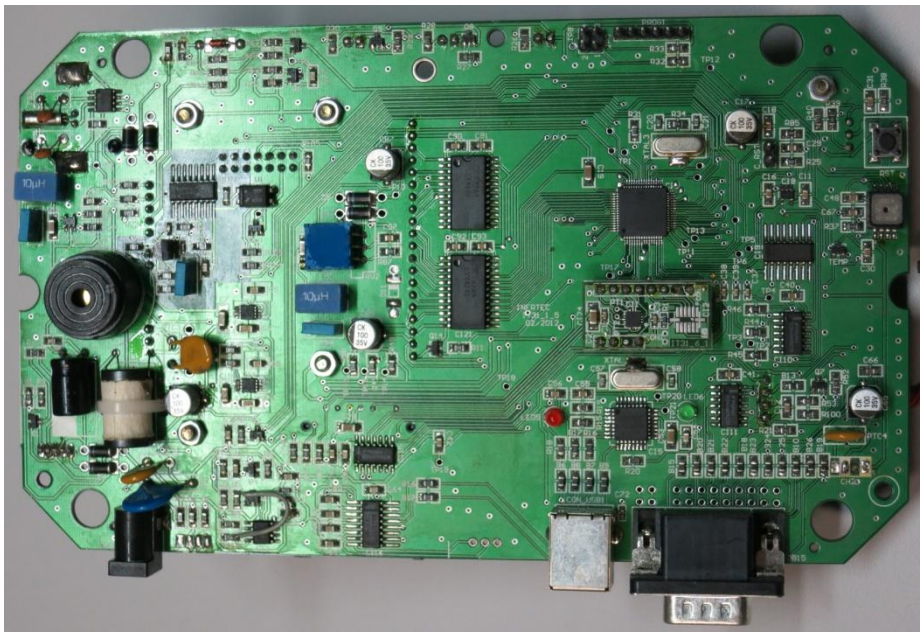


Figura 25 - Placa com dsPIC: Vista inferior

A Figura 26 mostra com mais detalhes o circuito da fonte de alimentação com os reguladores LM2575 e MCP1252.

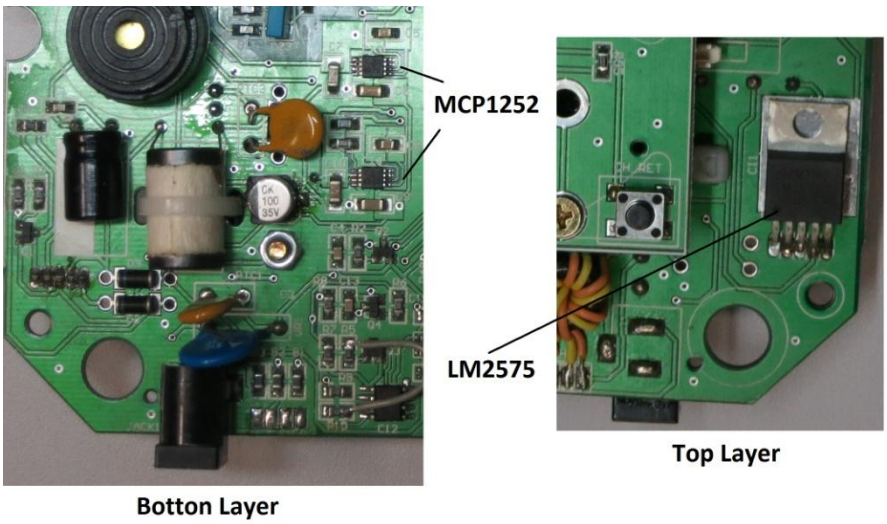


Figura 26 - Circuito da fonte de alimentação (vistas superior e inferior)

A Figura 27 mostra detalhes do circuito do conversor USB FT232.

Figura 27 - Circuito do conversos USB FT232

Depois de adicionar o quadricóptero no MSFS, programar o FVMS V2.0, projetar, desenvolver e instalar o *hardware* externo com o dsPIC, foram realizados alguns testes de estabilidade. O primeiro deles, foi repetir o teste de Monte Carlo, conforme realizado por Louali et al. (2011). Na sequência foi realizado um teste simples de estabilidade com a aeronave pairando sem distúrbios no sistema. Utilizando um *joystick* USB, introduzimos alguns distúrbios para confirmarmos a eficácia do controlador e do sistema SIL e HIL como um todo. Por fim, foram realizadas alterações nas condições meteorológicas no MSFS para ver como o quadricóptero se comportava.

5.2. Teste Monte Carlo

O teste de Monte Carlo é um método estatístico que se baseia em amostragens aleatórias massivas para obter resultados numéricos, isto é, repetindo sucessivas simulações um elevado número de vezes, para calcular probabilidades heurísticas (Wikipedia, 2014). Este teste foi utilizado por Louali et al. (2011) num estudo sobre avaliação da performance do MSFS como sistema de tempo real. O teste consiste em realizar um número grande de leituras dos sensores, neste caso N=1000, verificar o tempo gasto para essas leituras e, com a média aritmética dos valores obtidos pelas equações (11) e (12), calcular a frequência média (f_{med}) de atualização dessas variáveis. A acurácia dessa medida será calculada através do desvio padrão (σ), dado pela equação (13).

$$T_m = \frac{1}{N}\sum_{i=1}^{N} T_i \tag{11}$$

$$f_{med} = \frac{1}{T_m} \tag{12}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_i - T_m)^2} \tag{13}$$

A quantidade de dados lidas da memória através da DLL FSUIPC é de 20000 bytes. Nos testes realizados por Louali et al. (2011), a quantidade de dados lidos foi inferior, no entanto, julgamos que a leitura de 20000 bytes neste teste seria mais condizente com a realidade deste trabalho. Conforme descrito no mesmo trabalho, os autores obtiveram resultados diferentes quando o simulador estava minimizado ou maximizado na tela do computador, portanto, julgamos prudente testar nas mesmas condições. Deve-se ressaltar, no entanto, que uma das maiores vantagens de se utilizar o MSFS para simulação é poder visualizar em tempo real o que acontece com a aeronave simulada.

Um outro fator importante, também ressaltado por Louali et al. (2011), refere-se à máquina e ao sistema operacional utilizados nos testes. A Tabela 18 resume os resultados no teste de Monte Carlo, obtidos neste trabalho e por Louali et al. (2011).

Tabela 18 - Resultados do teste Monte Carlo

Condição	T_m **[mS]**	f_{med} **[Hz]**	σ **[mS]**	T_{ref} **[mS]**	f_{ref} **[Hz]**	σ_{ref} **[mS]**
				(LOUALI *et al.*, 2011)		
MSFS maximizado	49,5610	20,1771	5,9612	282.5981	3.5398	26.2878
MSFS minimizado	19,4060	51,5305	11,3589	13.3310	75.0134	0.0716

Analisando esses resultados conclui-se que a frequência obtida neste trabalho quando o MSFS está minimizado é bem menor do que a frequência obtida por Louali et

al. (2011). No entanto, o resultado obtido quando o *software* ficou maximizado na tela foi bem superior ao obtido no mesmo trabalho. A justificativa para isso está nas características do computador e do sistema operacional utilizados. A Tabela 19 resume as características dos computadores utilizados nos dois testes.

Tabela 19 - Características dos computadores

	PC1 (LOUALI *et al.*, 2011)	**PC2** (utilizado na realização deste trabalho)
O.S.	Windows XP	Windows 7
Processador	Intel Xeon 8-core 2GHz	Intel Extreme i7-3970X 3.5GHz
Memória RAM	2,0 GB	32,0 GB
Hard Disk	140 GB	1 TB
Placa gráfica	NVIDIA Quadro FX4600 768 MB GDDR3	NVIDIA Geforce GT640 2 GB DDR3
Monitor de vídeo	LCD, 60 Hz	LCD, 60 Hz

Nos outros testes detalhados na sequência deste trabalho, o trabalho operacional executado pelo FVMS não se limitou apenas a ler as variáveis do MSFS, como no teste de Monte Carlo. O ciclo total consistiu em: ler as variáveis, executar o controle (no FVMS para o SIL e no dsPIC para o HIL) e atuar nas superfícies de controle (*elevator* e *aileron*). Portanto, os resultados de frequência e desvio padrão neste caso foram diferentes dos obtidos na Tabela 18. No entanto, a diferença entre o MSFS estar ou não minimizado continuou sendo significativa. Portanto, todos os testes realizados foram feitos em 4 condições diferentes: SIL com MSFS maximizado, SIL com MSFS minimizado, HIL com MSFS maximizado e HIL com MSFS minimizado. A Tabela 17 mostra os resultados do Período Médio (T_m), Frequência Média (f_{med}) e Desvio Padrão (σ) para cada um desses casos. Em todos os testes realizados nessas condições, esses resultados foram praticamente os mesmos, conforme relacionado na Tabela 16 e na Tabela 17.

Baseado nesses resultados, nota-se a semelhança dos resultados do SIL e do HIL com o MSFS maximizado. O tempo decorrido entre o início da comunicação e o término

da resposta é de 6,40mS, dos quais 597uS correspondem ao tempo de execução do controle propriamente dito. A Figura 28, correspondente ao gráfico gerado pelo osciloscópio digital colocando-se as pontas de prova nos pontos USB_TX e USB_RX da placa (Figura 13), confirma essas informações.

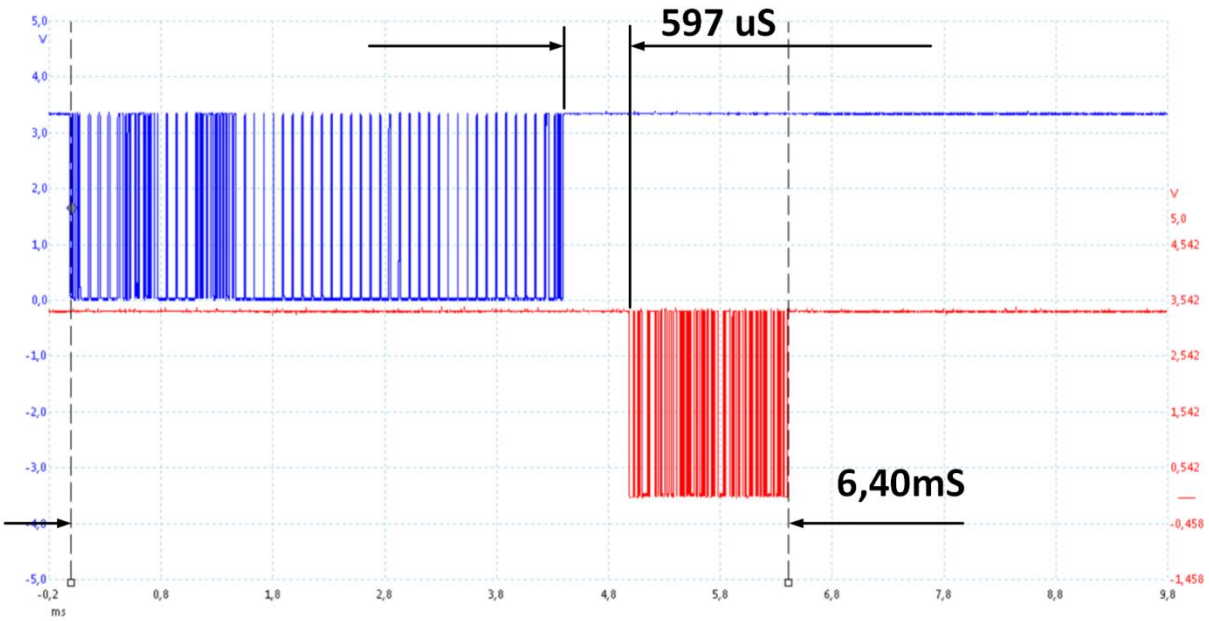


Figura 28 - Sinais USB_TX e USB_RX osciloscópio

O tempo de comunicação, portanto, representa 3,77% do período do SIL maximizado e 9,18% do SIL minimizado. Ou seja, o impacto dele no SIL maximizado é muito menor do que no minimizado.

5.3. Comportamento do sistema durante a decolagem

Neste teste, o quadricóptero decola até uma altitude entre 20 e 50m. O procedimento de decolagem foi feito manualmente com o *joystick*. Como não existe controle de posicionamento, é difícil garantir que a altura seja a mesma em todas as situações.

5.3.1. Decolagem SIL com o MSFS maximizado

O gráfico mostrado na Figura 29 mostra a variação dos ângulos de *pitch* e *roll* durante a decolagem. Note que, neste teste especificamente, quando não ocorreu variação do ângulo de rolagem. Somente o ângulo de arfagem é que variou um pouco durante os primeiros 5s.

Compare esse mesmo período nos gráficos das figuras Figura 30 e Figura 31. O primeiro deles mostra o sinal de controle recebido do controlador e o segundo a respectiva atuação no *elevator*. Note que os sinais de controle dos motores 2 e 4, responsáveis pela correção do ângulo de rolagem, praticamente não existem. O mesmo acontece na atuação do *aileron*.

A Figura 32 e a Figura 33 mostram o gráfico da frequência de atuação e o respectivo período. Este período é o tempo decorrido entre duas execuções consecutivas do controle. Em ambos os gráficos, a curva não tracejada e em vermelho representa o valor médio (média aritmética), gerado simultaneamente com as aquisições, execução e atuação do controle. Os valores da frequência variaram entre 5,8Hz e 6,4Hz, sendo que este último ocorria, em média, a cada 9 ciclos. O valor médio 5,89Hz, portanto, se mostra coerente com a situação. Com relação ao período, os valores variaram de 156ms a 172ms, com média de 169,68ms.

O gráfico da Figura 34 mostra apenas a variação da altitude durante o ensaio. Como o controlador está controlando apenas os ângulos de rolagem e arfagem, este gráfico não nos retorna informações quantitativas importantes. No entanto, ele pode mostrar de forma empírica, eventuais instabilidades.

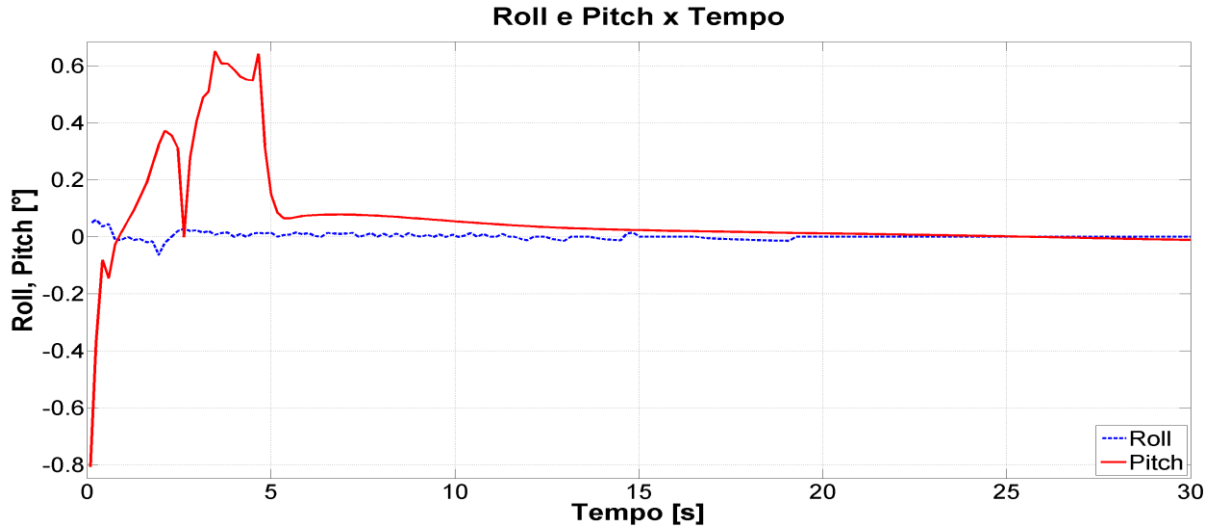


Figura 29 - SIL, MSFS maximizado, Gráfico Roll e Pitch

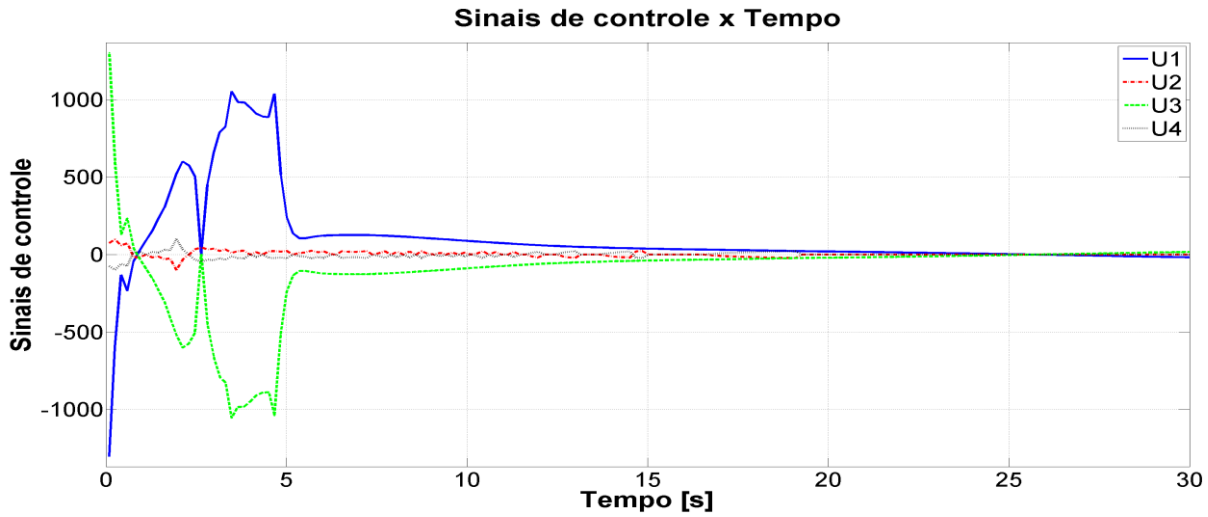


Figura 30 - SIL, MSFS maximizado, Gráfico Sinais de controle

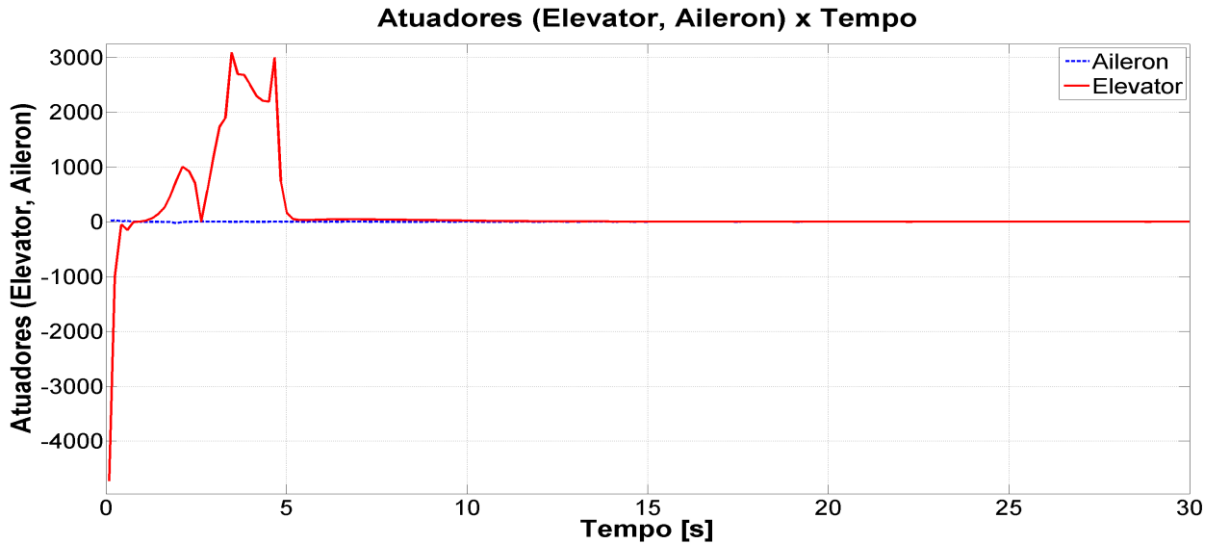


Figura 31 - SIL, MSFS maximizado, Gráfico Atuadores (Elevator, Aileron)

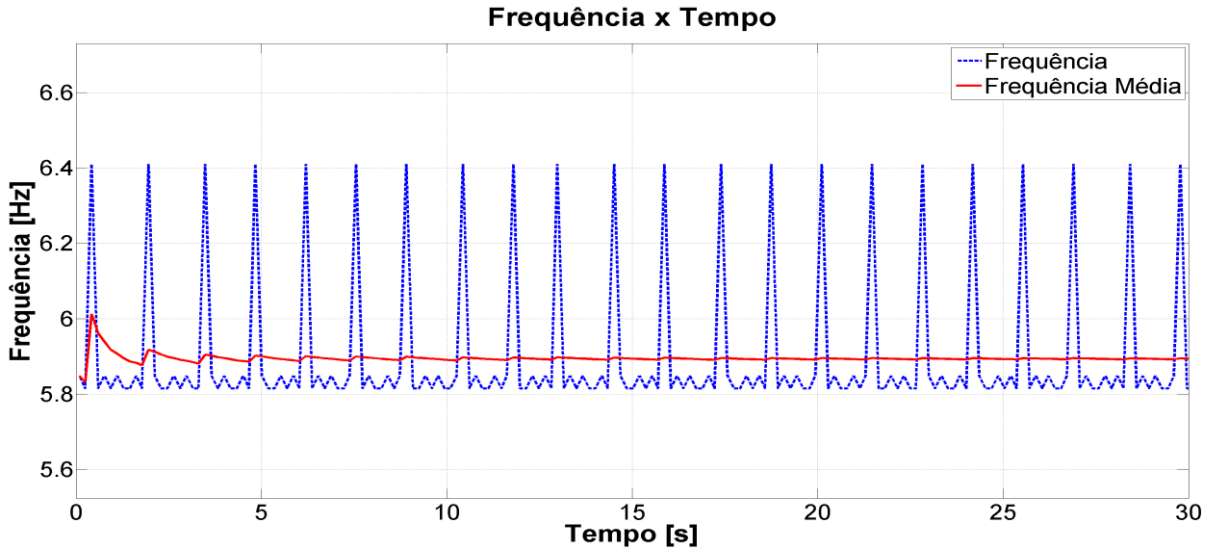


Figura 32 - SIL, MSFS maximizado, Gráfico Frequência

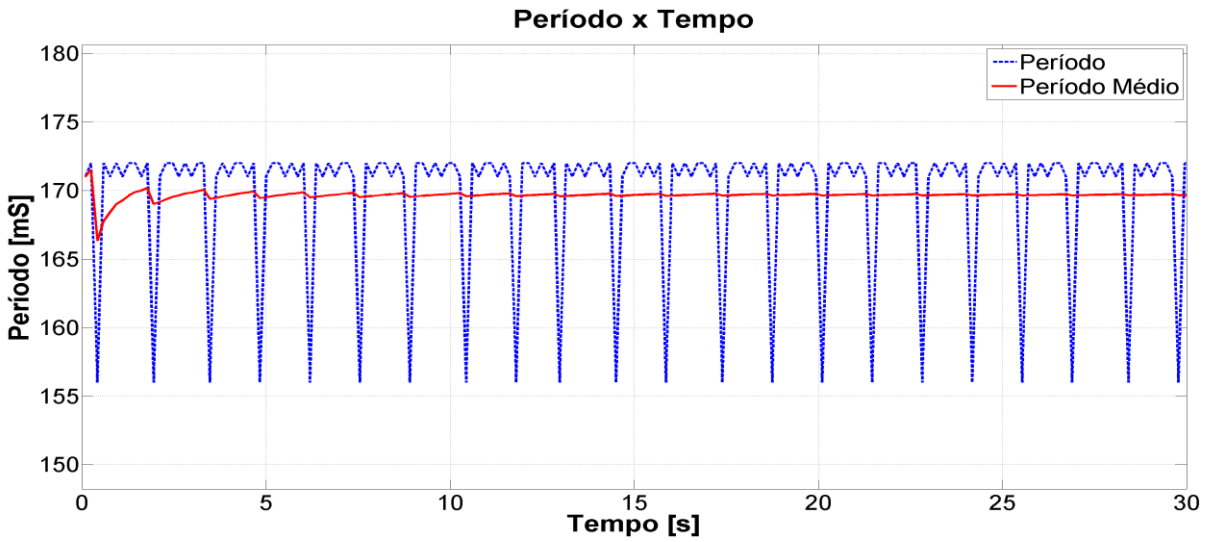


Figura 33 - SIL, MSFS maximizado, Gráfico Período

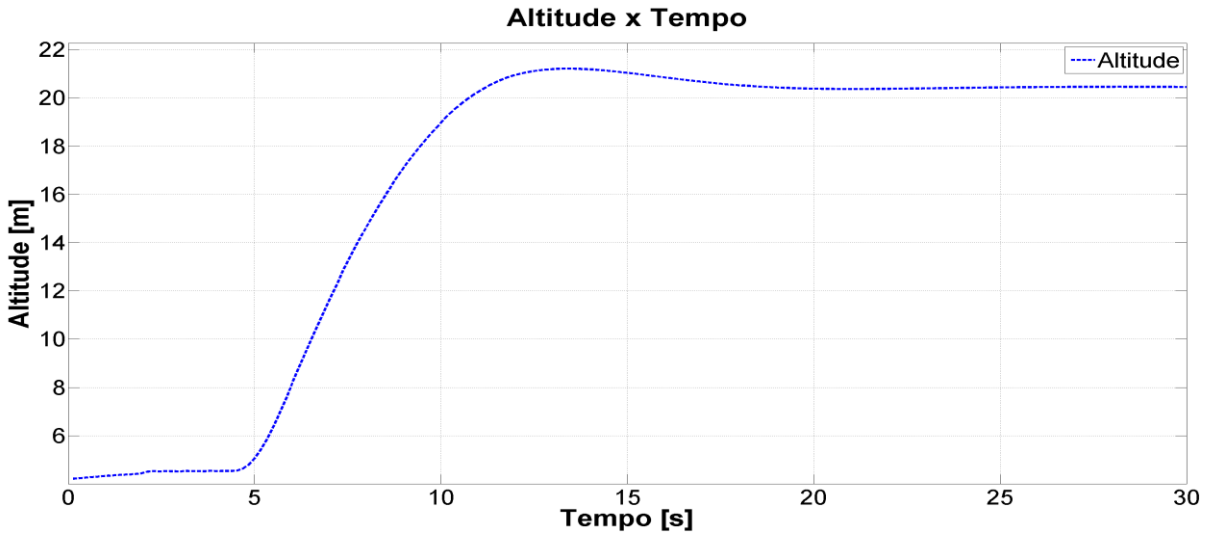


Figura 34 - SIL, MSFS maximizado, Gráfico da Altitude

5.3.2. Decolagem SIL com o MSFS minimizado

Os gráficos da Figura 35, da Figura 36 e da Figura 37 mostram a variação dos ângulos de atitude, o sinal de controle e a atuação nas superfícies de controle. Esses gráficos deixam evidente a diferença na questão da frequência de atuação do controle. Note que os sinais de controle reagem de forma bem mais rápida.

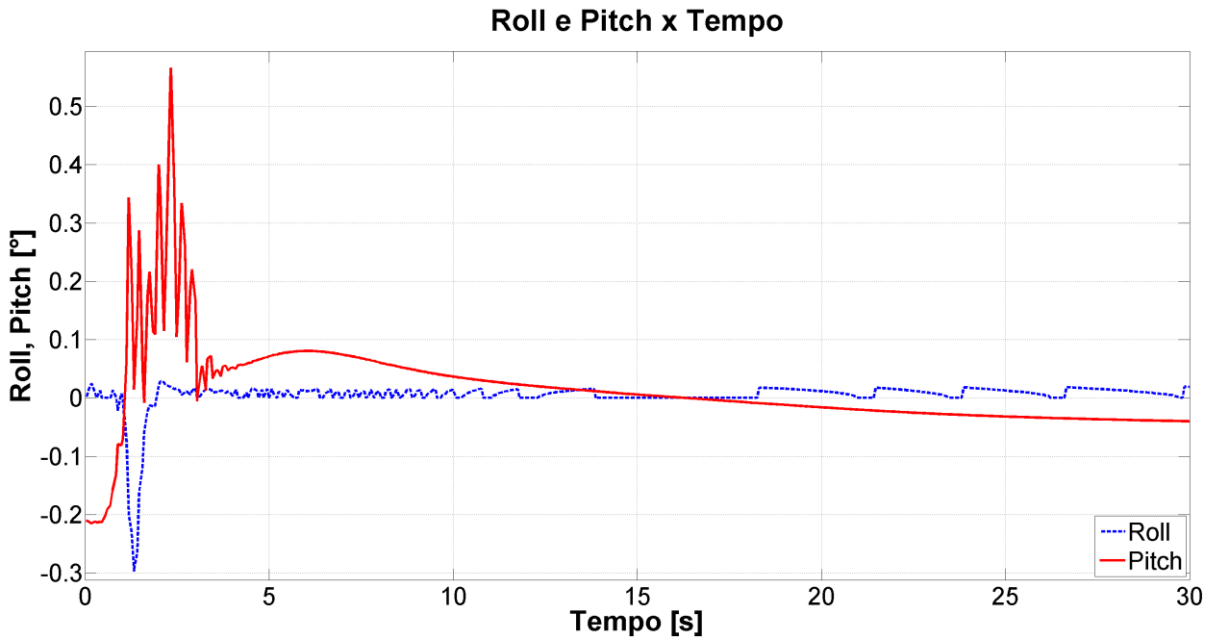


Figura 35 - SIL, MSFS minimizado, Gráfico Roll e Pitch

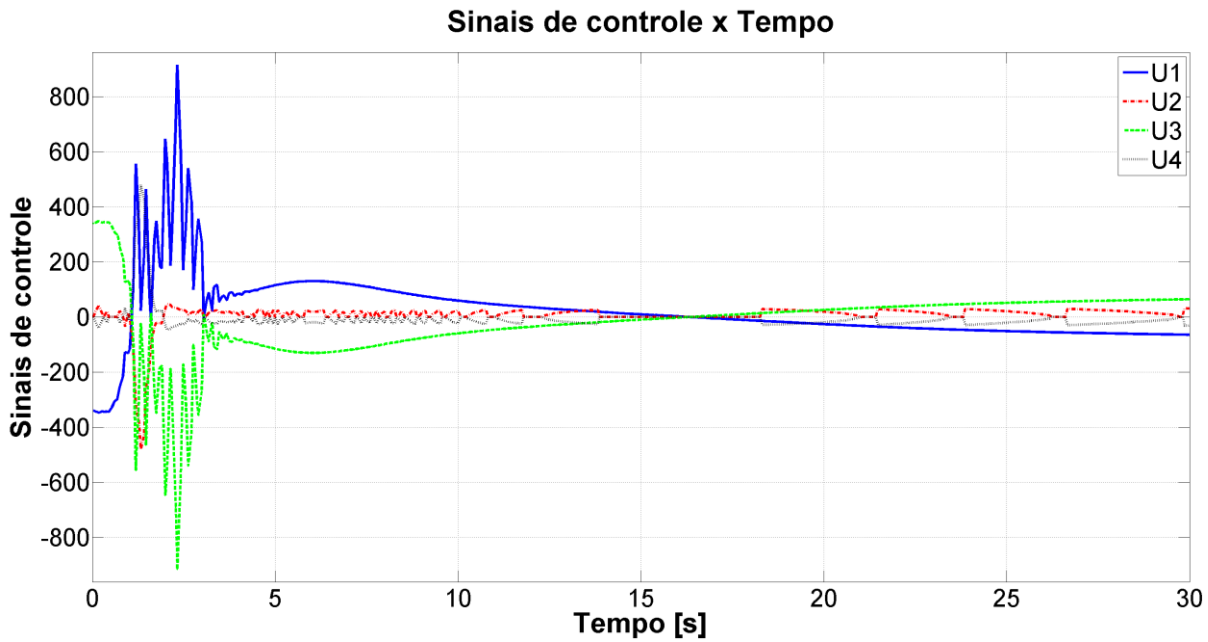


Figura 36 - SIL, MSFS minimizado, Gráfico Sinais de controle

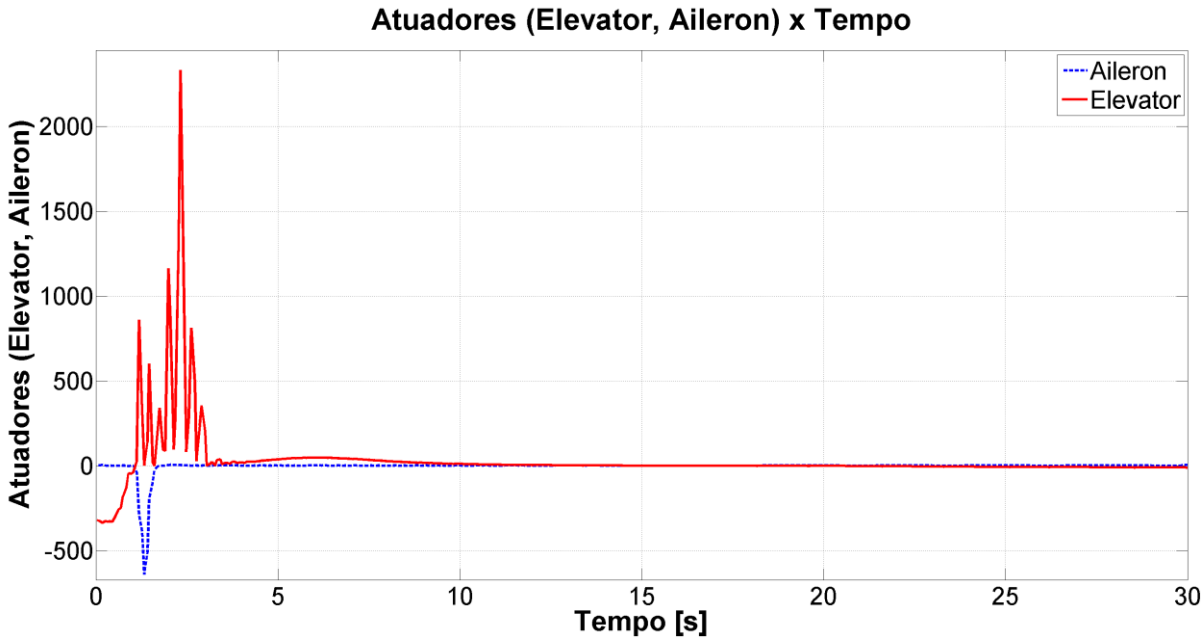


Figura 37 - SIL, MSFS minimizado, Gráfico Atuadores (Elevator e Aileron)

Nos gráficos da Figura 38 e da Figura 39, pode-se notar como a velocidade de atuação do controle aumentou, melhorando com isso a eficiência do sistema. O valor mínimo de frequência foi de 12,82Hz, o máximo 21,28Hz e o médio 14,35Hz. Vê-se também que esse valor máximo obtido deve ser descartado, pois representa uma amostra apenas. Com relação ao período, o valor mínimo obtido foi de 47ms, o máximo 78ms e o médio 69,71ms.

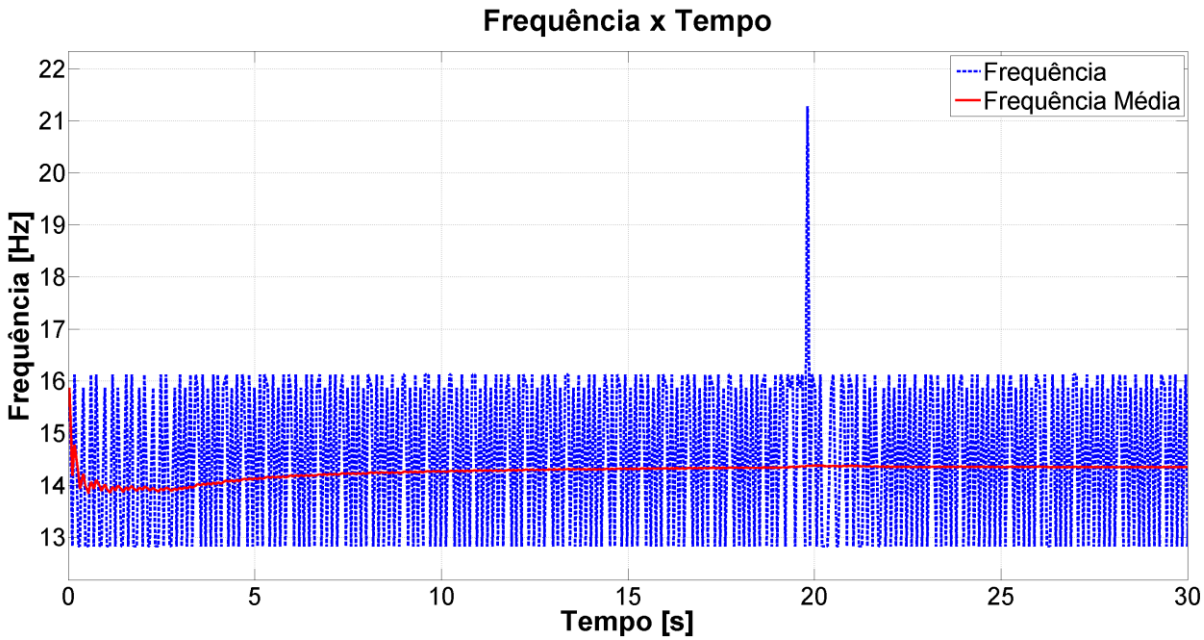


Figura 38 - SIL, MSFS minimizado, Gráfico Frequência

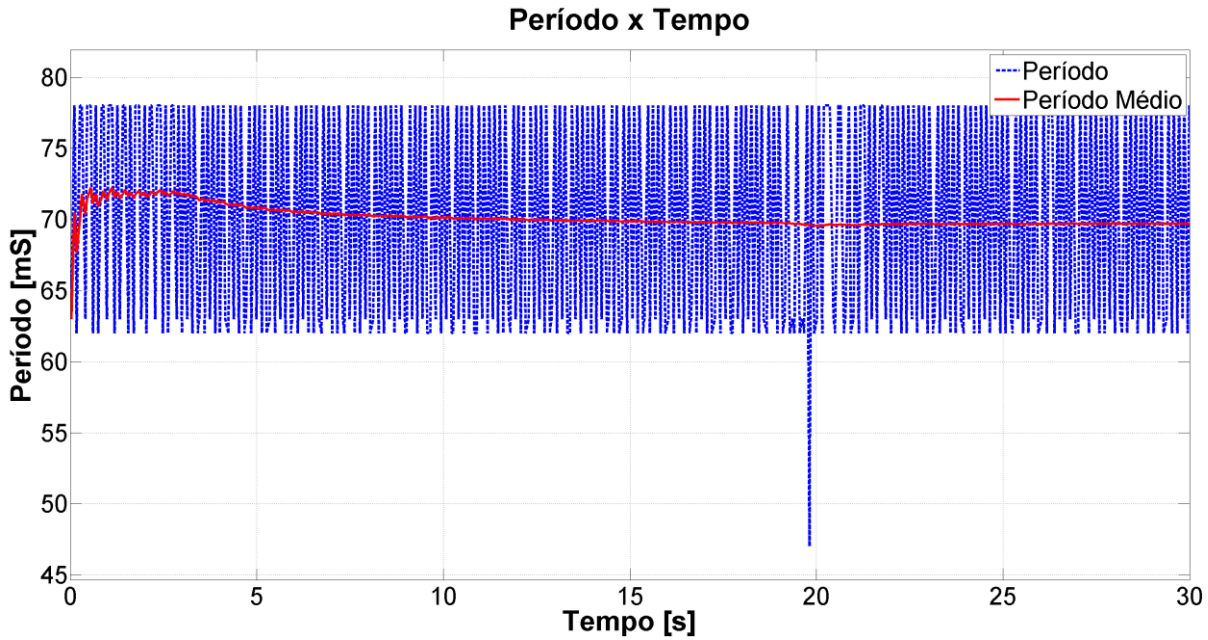


Figura 39 - SIL, MSFS minimizado, Gráfico do Período

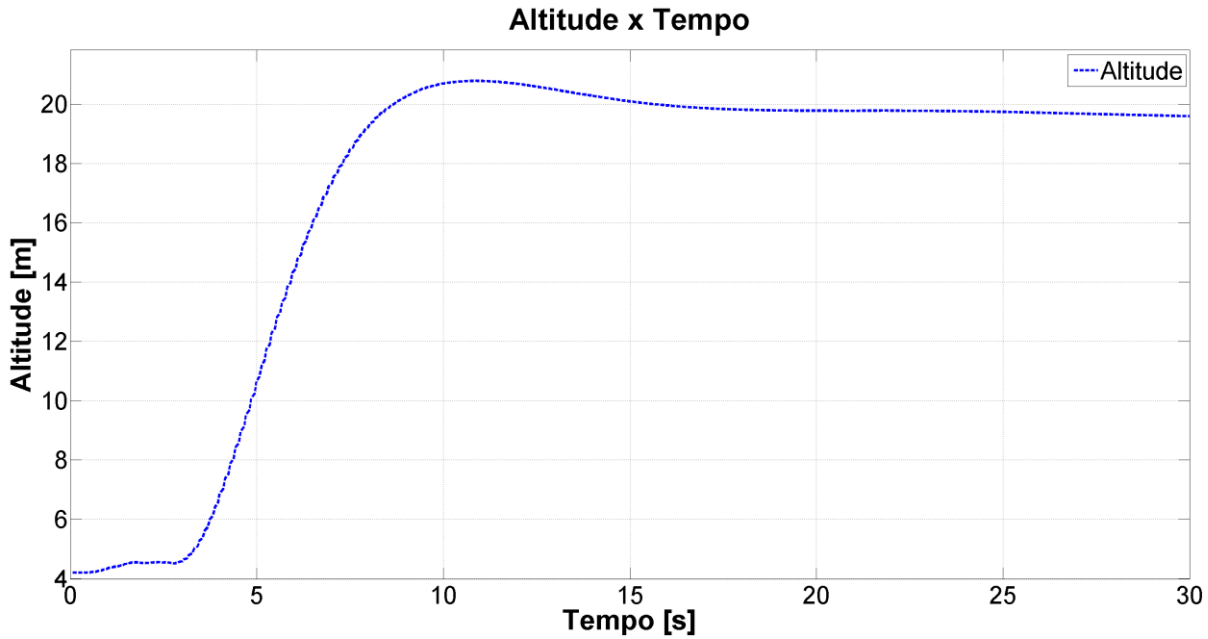


Figura 40 - SIL, MSFS minimizado, Gráfico da Altitude

5.3.3. Resultados para o HIL com o MSFS maximizado

Os gráficos da Figura 41, da Figura 42 e da Figura 43 mostram a variação dos ângulos de atitude, os sinais de controle e a atuação nas superfícies de controle, respectivamente. Como era de se esperar, a velocidade de atuação ficou coerente com a do teste do item 5.3.1.

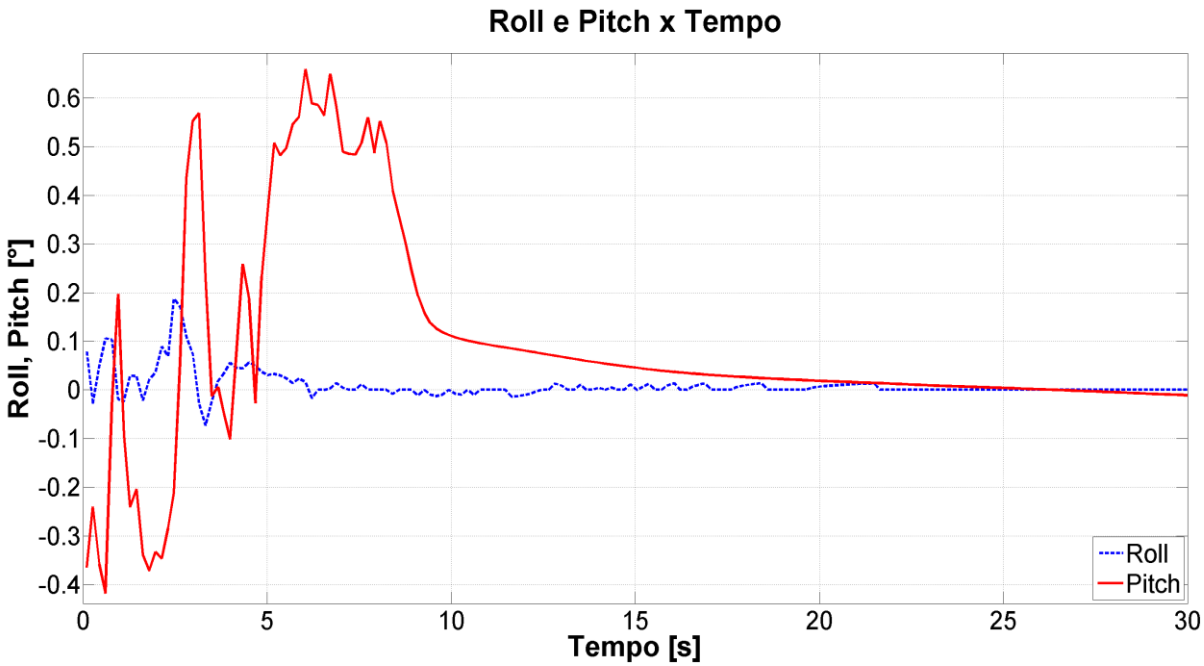


Figura 41 - HIL, MSFS maximizado, Gráfico Roll e Pitch

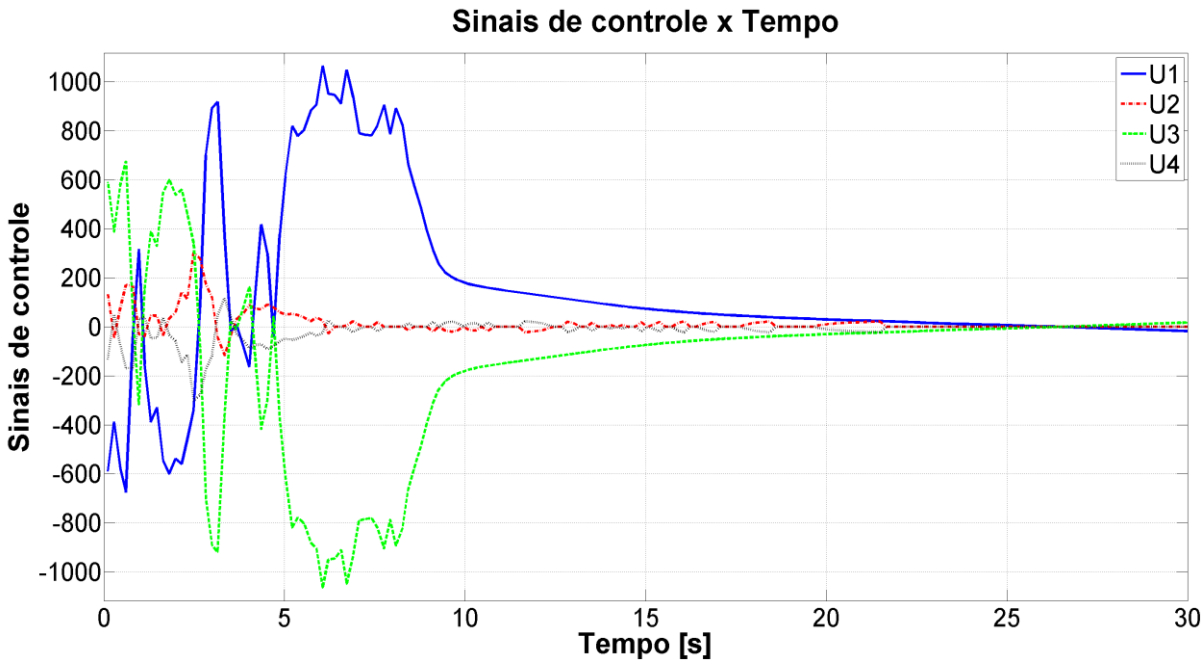


Figura 42 - HIL, MSFS maximizado, Gráfico dos sinais de controle

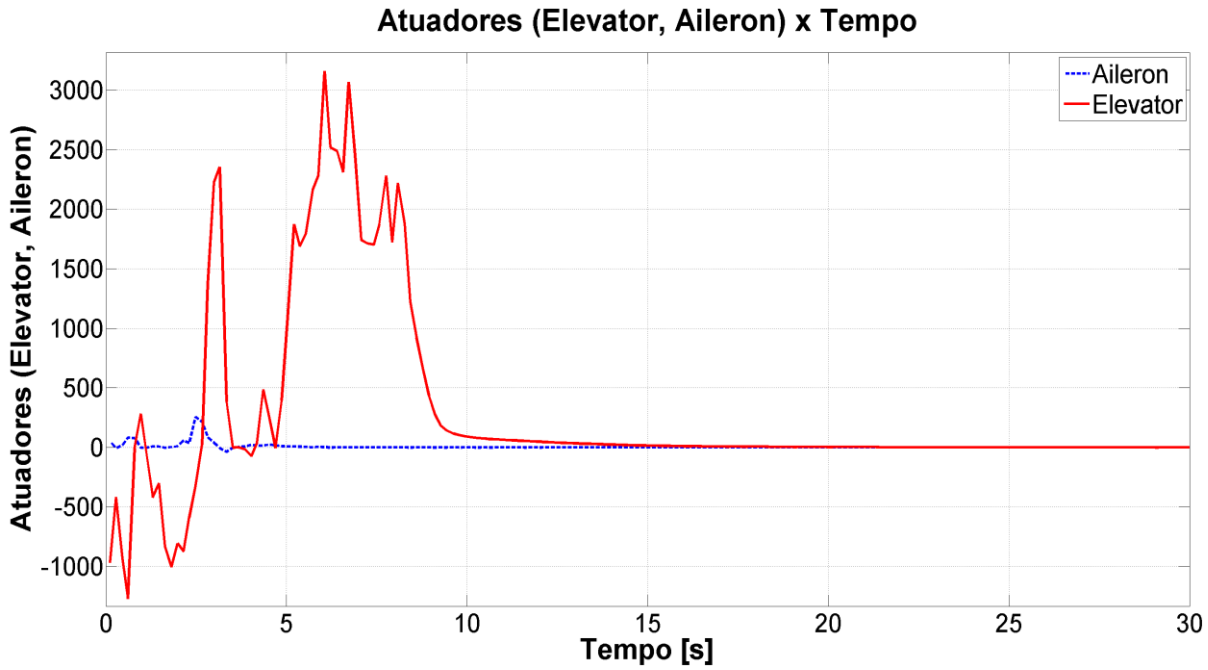


Figura 43 - HIL, MSFS maximizado, Gráfico dos atuadores (Elevator e Aileron)

Nos gráficos da Figura 44 e da Figura 45, pode-se notar como a velocidade de atuação do controle voltou a ser menor. O valor mínimo de frequência foi de 5,81Hz, o máximo 6,41Hz e o médio 5,89Hz. Com relação ao período, o valor mínimo obtido foi de 156ms, o máximo 172ms e o médio 169,75ms. Todos os testes na arquitetura HIL e com o MSFS maximizado foram os mais lentos.

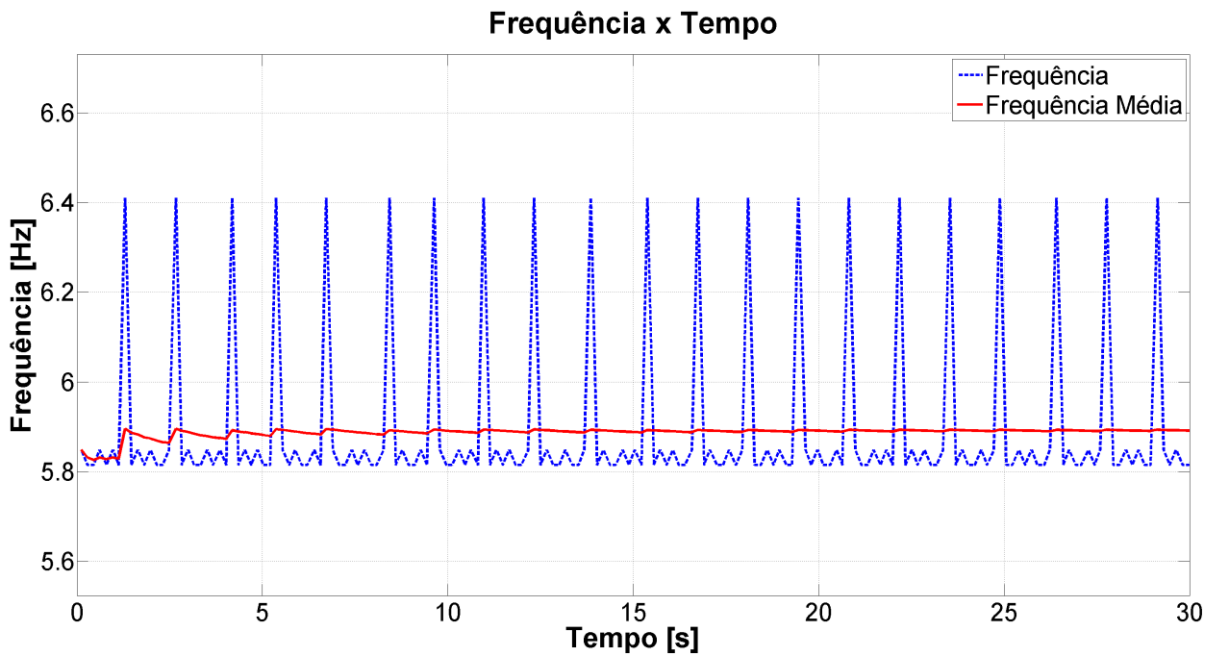


Figura 44 - HIL, MSFS maximizado, Gráfico da frequência

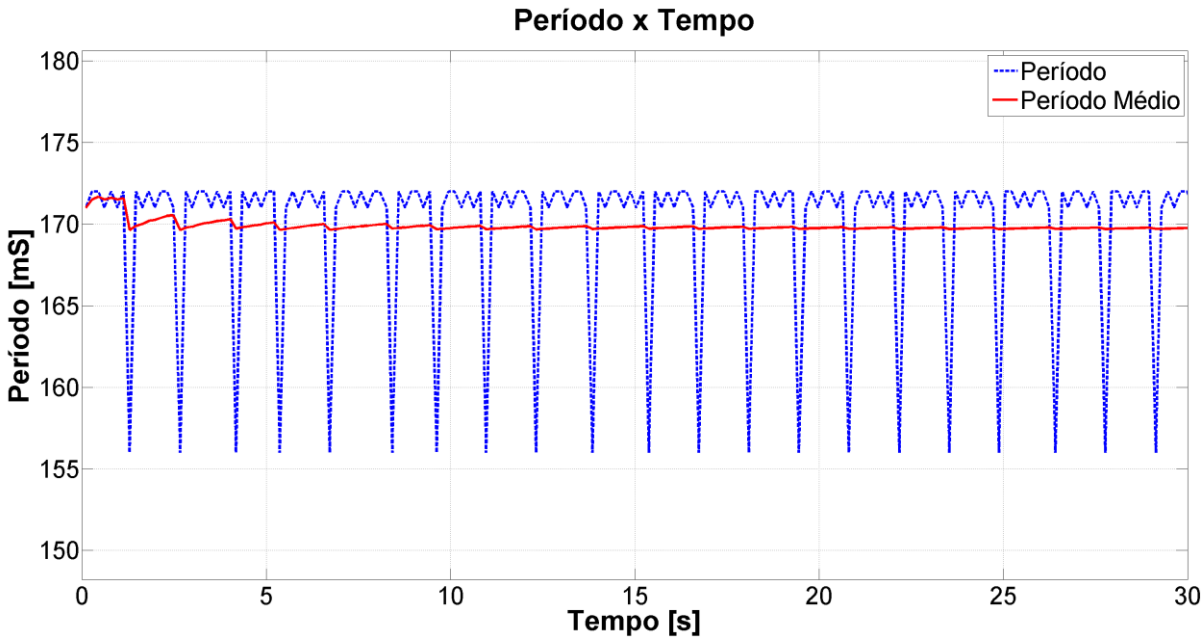


Figura 45 - HIL, MSFS maximizado, Gráfico do período

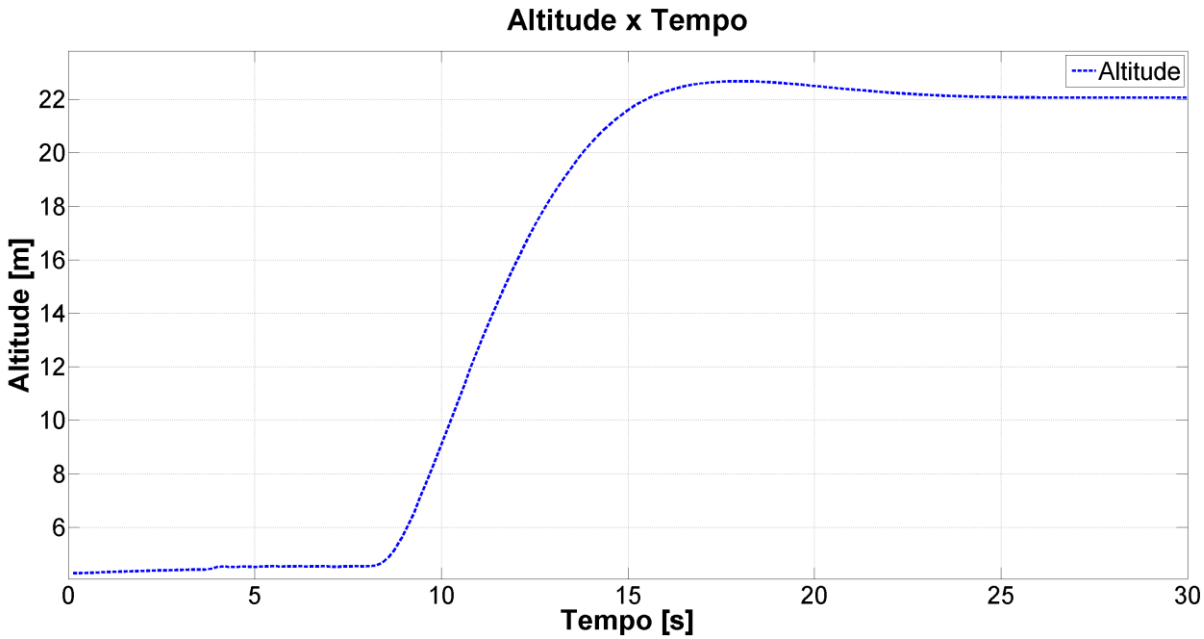


Figura 46 - HIL, MSFS maximizado, Gráfico da altitude

5.3.4. Resultados para o HIL com o MSFS minimizado

Os gráficos da Figura 47, da Figura 48 e da Figura 49 mostram a variação dos ângulos de atitude, os sinais de controle e a atuação nas superfícies de controle, respectivamente. Esses gráficos se assemelham aos obtidos no item 5.3.2.

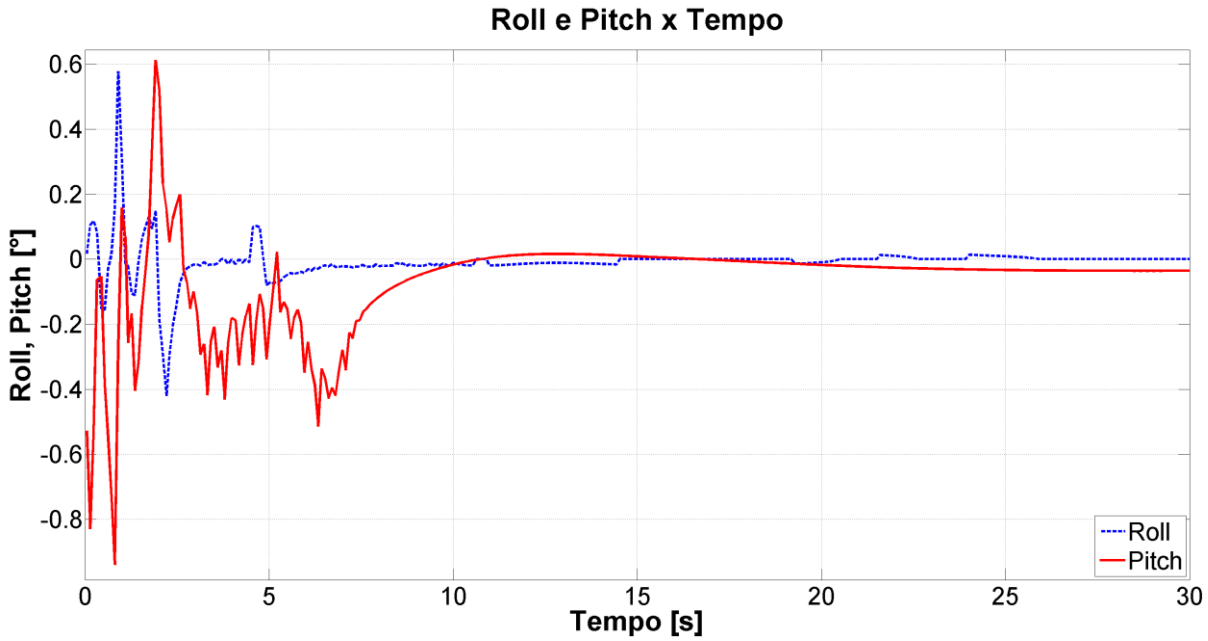


Figura 47 - HIL, MSFS minimizado, Gráfico Roll e Pitch

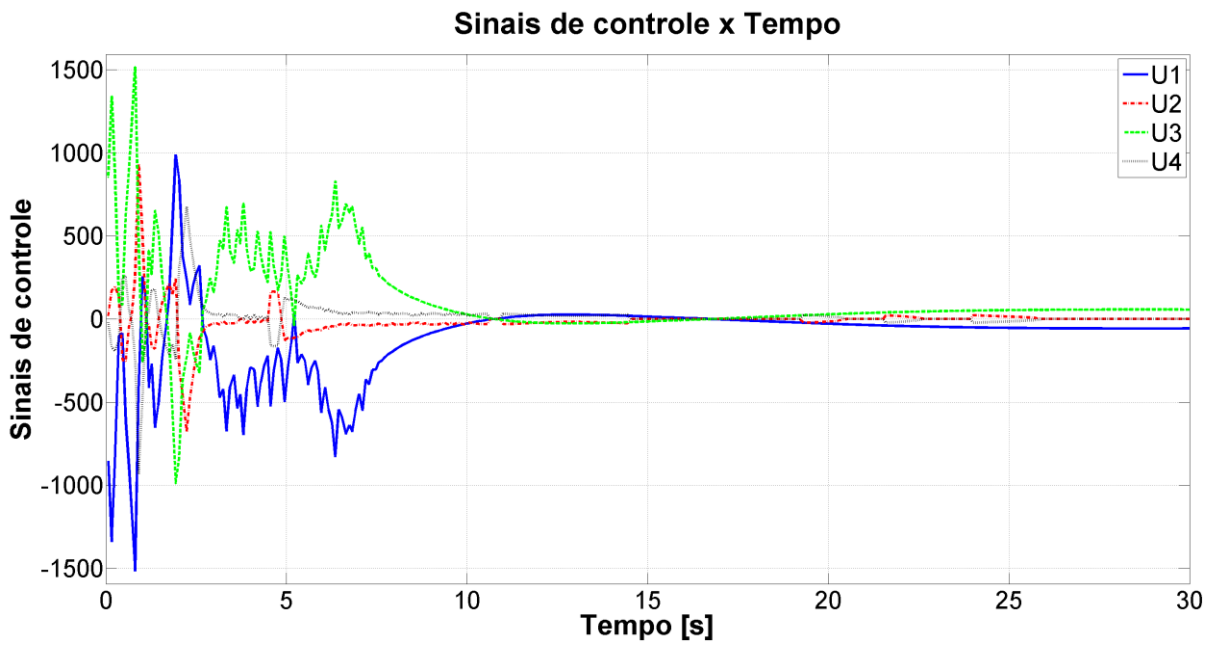


Figura 48 - HIL, MSFS minimizado, Gráfico Sinais de controle

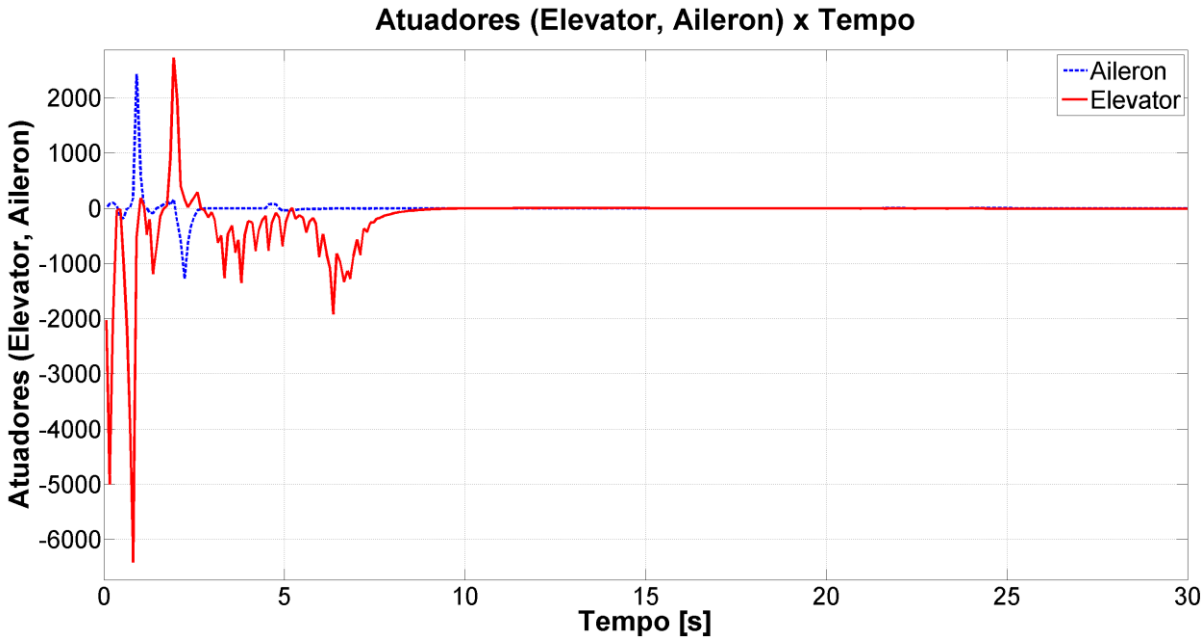


Figura 49 - HIL, MSFS minimizado, Gráfico dos Atuadores (Elevator e Aileron)

Nos gráficos da Figura 50 e da Figura 51, pode-se notar como a velocidade de atuação do controle se assemelha a do item 5.3.2. O valor mínimo de frequência foi de 9,17Hz, o máximo 16,13Hz e o médio 11,03Hz. Com relação ao período, o valor mínimo obtido foi de 62ms, o máximo 109ms e o médio 90,66ms. Todos os testes na arquitetura HIL e com o MSFS maximizado foram os mais lentos.

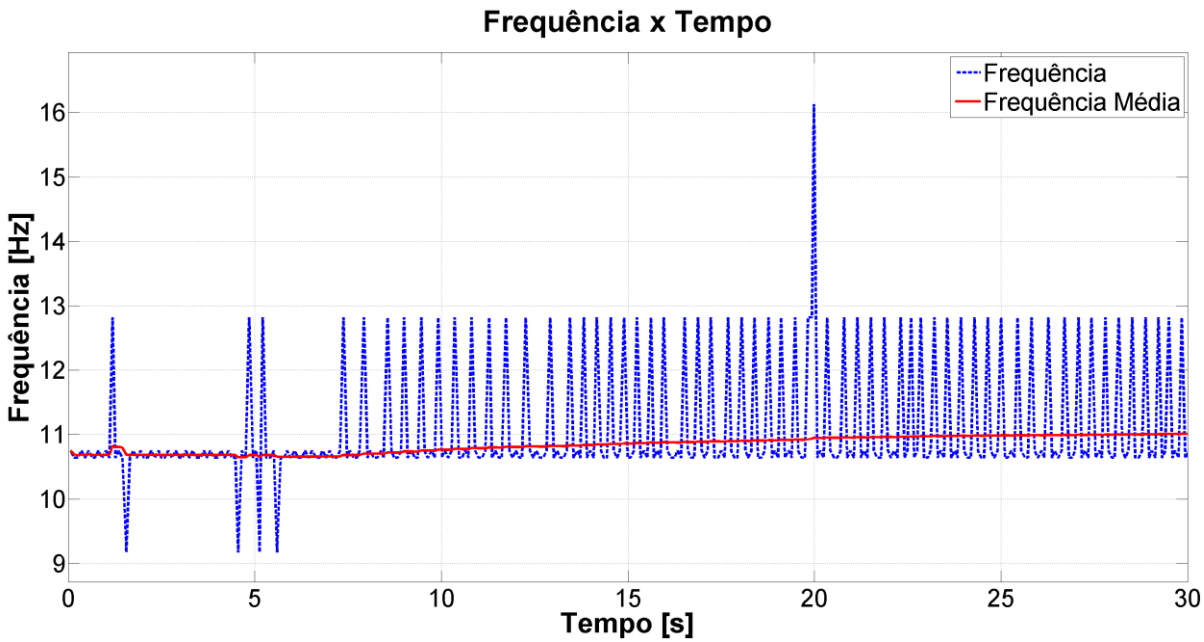


Figura 50 - HIL, MSFS minimizado, Gráfico da frequência

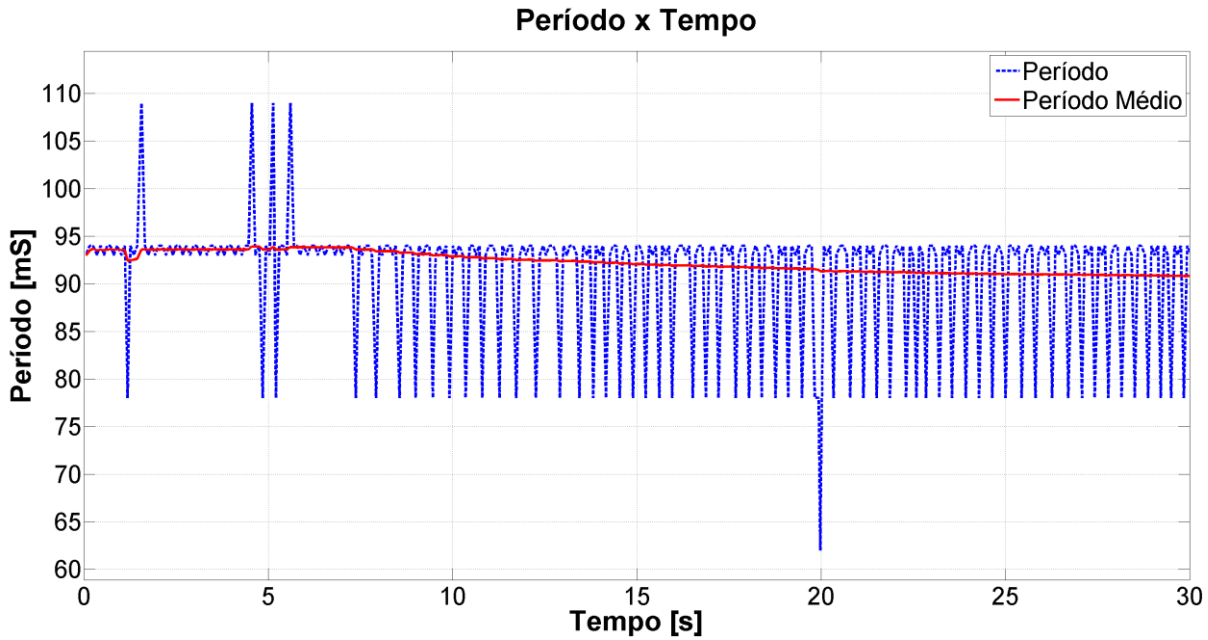


Figura 51 - HIL, MSFS minimizado, Gráfico do período

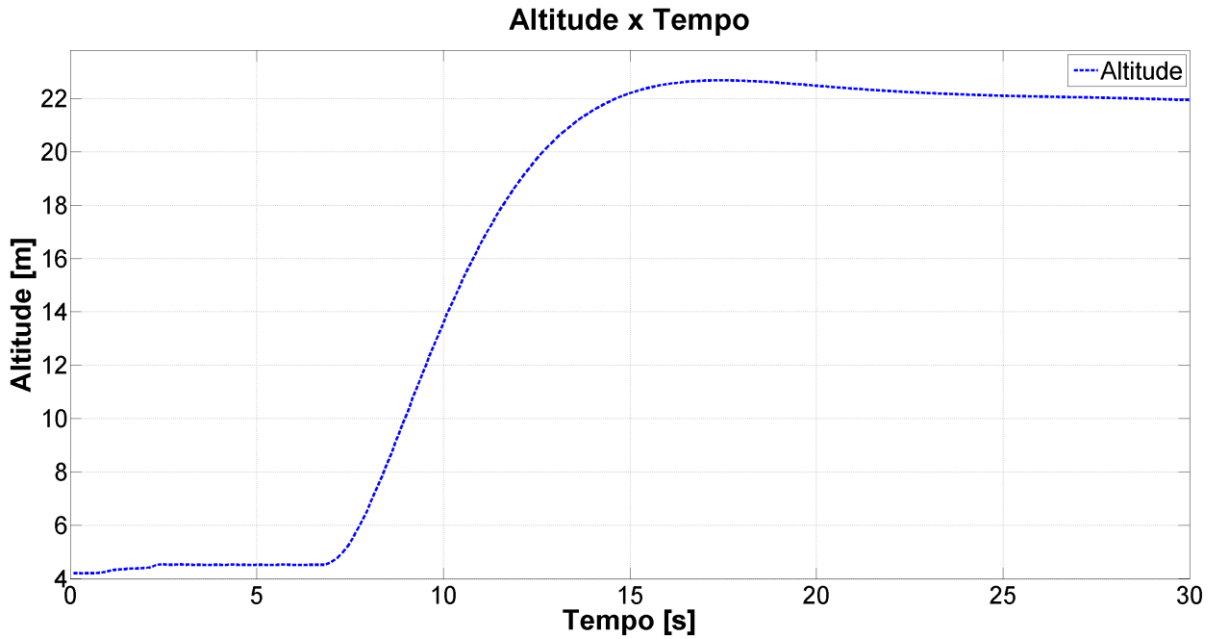


Figura 52 - HIL, MSFS minimizado, Gráfico da altitude

5.4. Teste de estabilidade com distúrbio simples aplicado no *joystick*.

No teste de estabilidade com distúrbio simples, o helicóptero decolou e, depois de estabilizado, sofreu solicitação de mudança dos ângulos de atitude nos 4 sentidos: *pitch* positivo, *pitch* negativo, *roll* positivo e *roll* negativo. O objetivo foi avaliar a reação do controlador nessas 4 situações. A solicitação de mudança dos ângulos foi realizada com o *joystick* conectado na porta USB do computador. A atuação do *joystick* acontece nas mesmas superfícies de controle (*aileron* e *elevator*) utilizadas pelo controlador, o que pode ser evidenciado nos gráficos dos atuadores.

De uma forma geral, os 3 primeiros gráficos ilustrados nas 3 primeiras figuras de cada seção (5.4.1, 5.4.2, 5.4.3 e 5.4.4), mostram a variação dos ângulos de atitude, dos sinais de controle e dos atuadores. O quarto gráfico mostra a variação da altitude durante o teste e os dois últimos a frequência e o período de execução do controle.

5.4.1. Resultados SIL com MSFS maximizado

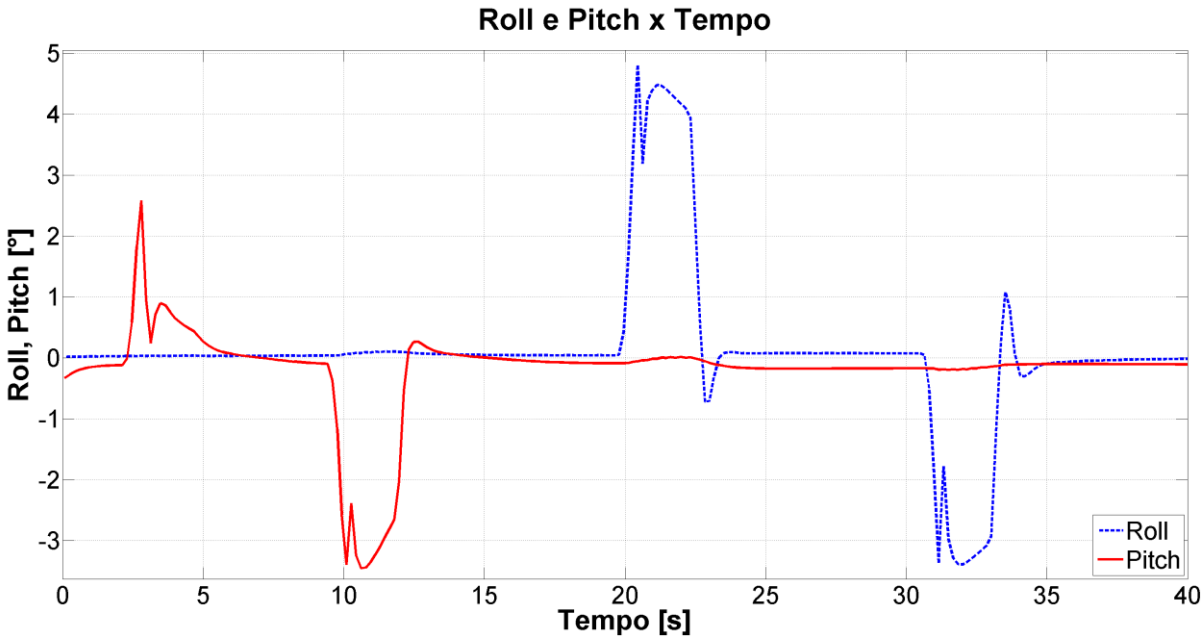


Figura 53 - SIL, MSFS maximizado, Gráfico Roll e Pitch

Na Figura 55, pode-se visualizar o efeito que a alteração no joystick gerou nos atuadores. No entanto, pode-se ver também, na Figura 54, a reação do controlador tentando corrigir a orientação do helicóptero.

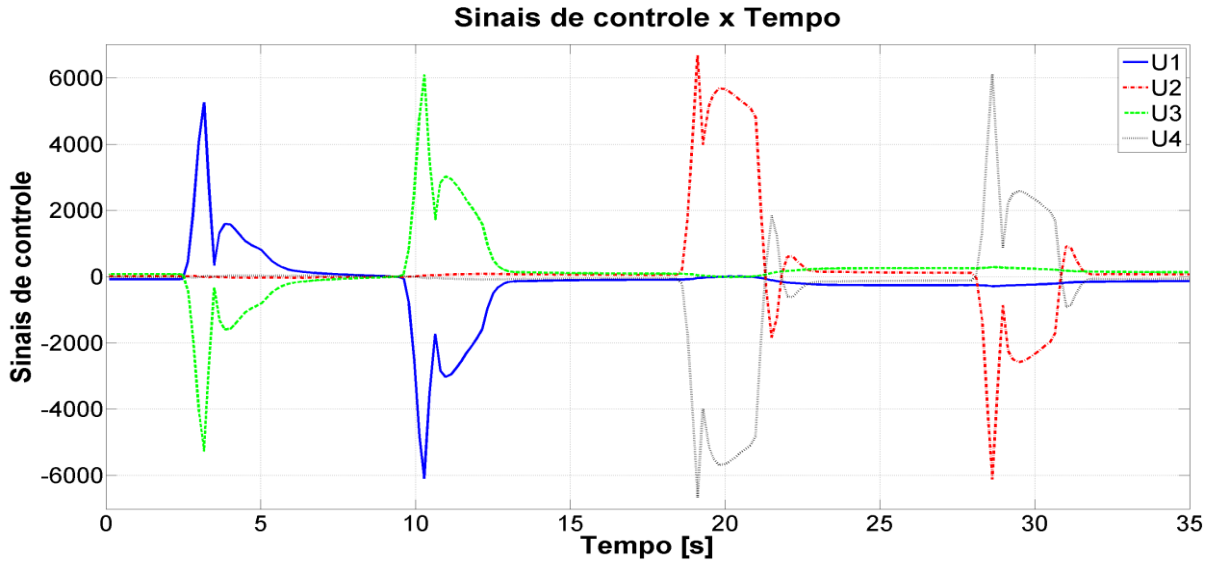


Figura 54 - SIL, MSFS maximizado, Gráfico Sinais de Controle

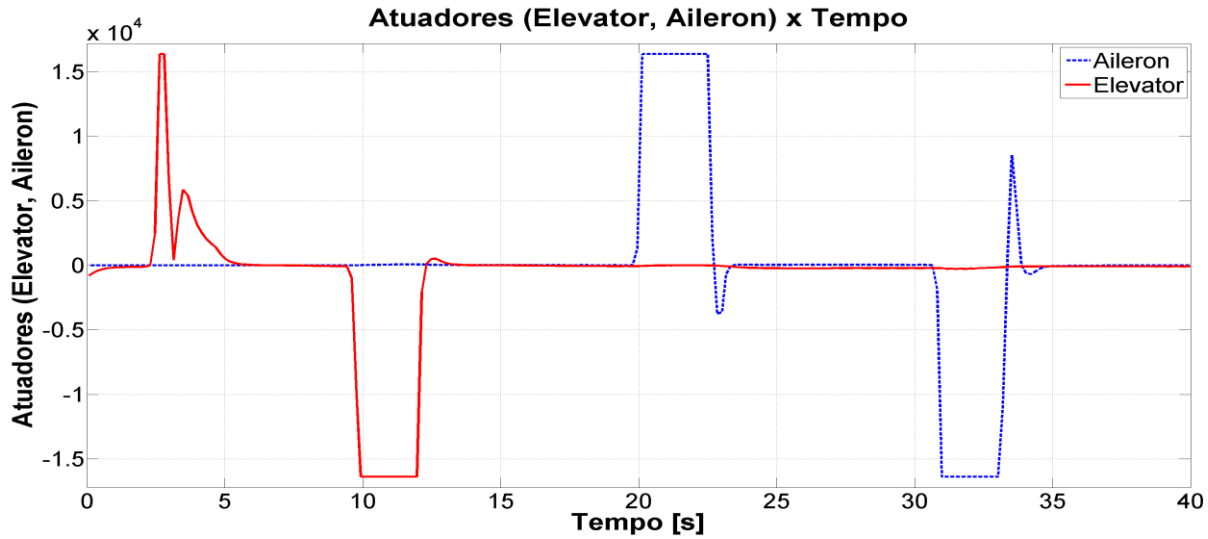


Figura 55 - SIL, MSFS maximizado, Gráfico de Atuação

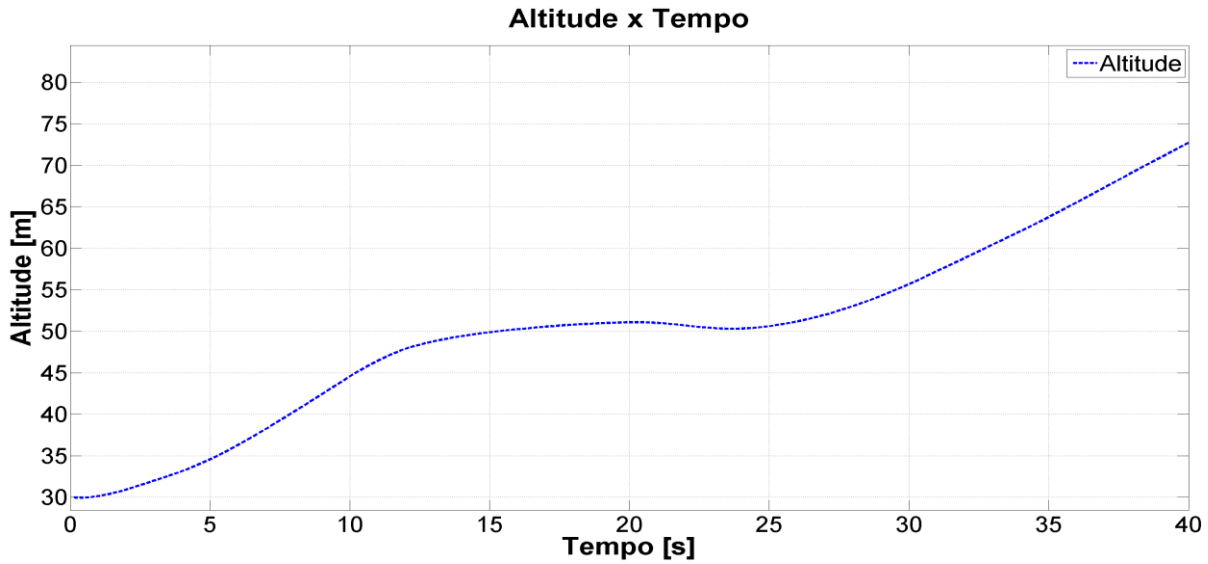


Figura 56 - SIL, MSFS maximizado, Gráfico da Altitude

Os valores mínimo, máximo e médio obtidos para a frequência de execução do controle foram respectivamente 5,81Hz, 6,41Hz e 5,89Hz (Figura 57). Os valores mínimo, máximo e médio obtidos para a período de execução do controle foram respectivamente 156ms, 172ms e 169,65ms (Figura 58).

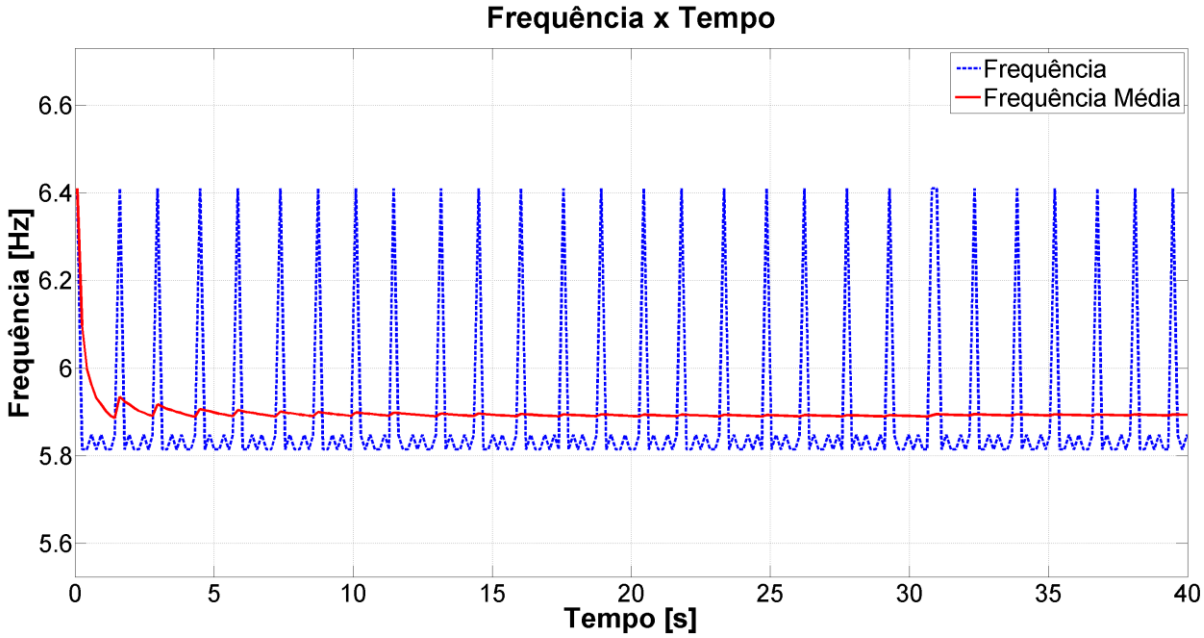


Figura 57 - SIL, MSFS maximizado, Gráfico da Frequência

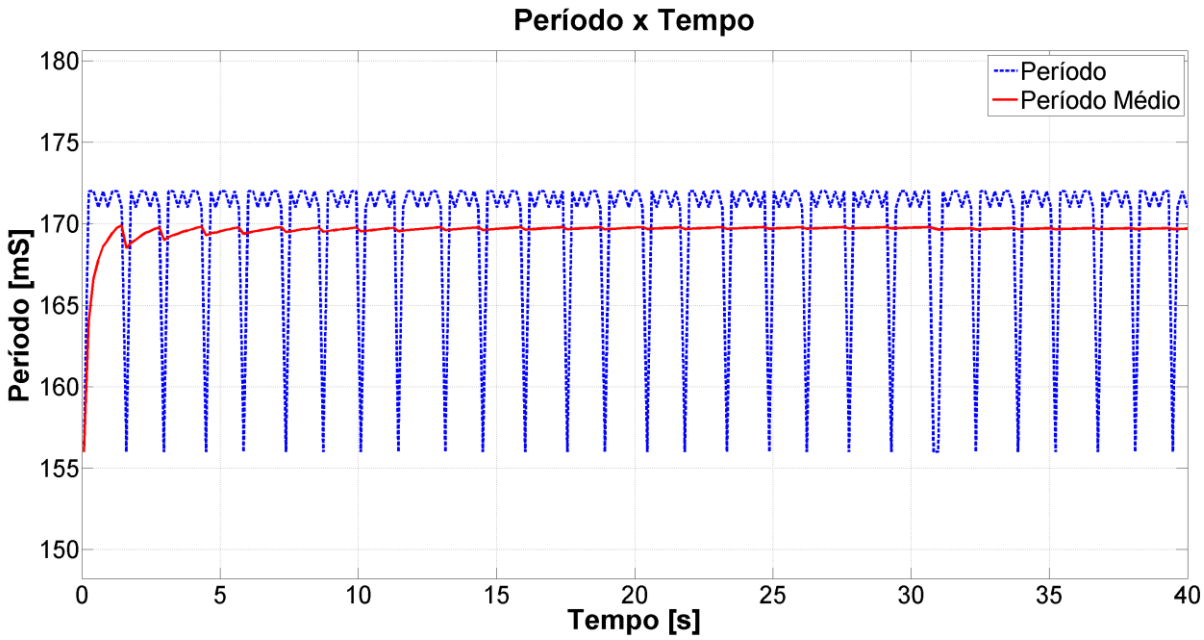


Figura 58 - SIL, MSFS maximizado, Gráfico do Período

5.4.2. Resultados SIL com MSFS minimizado

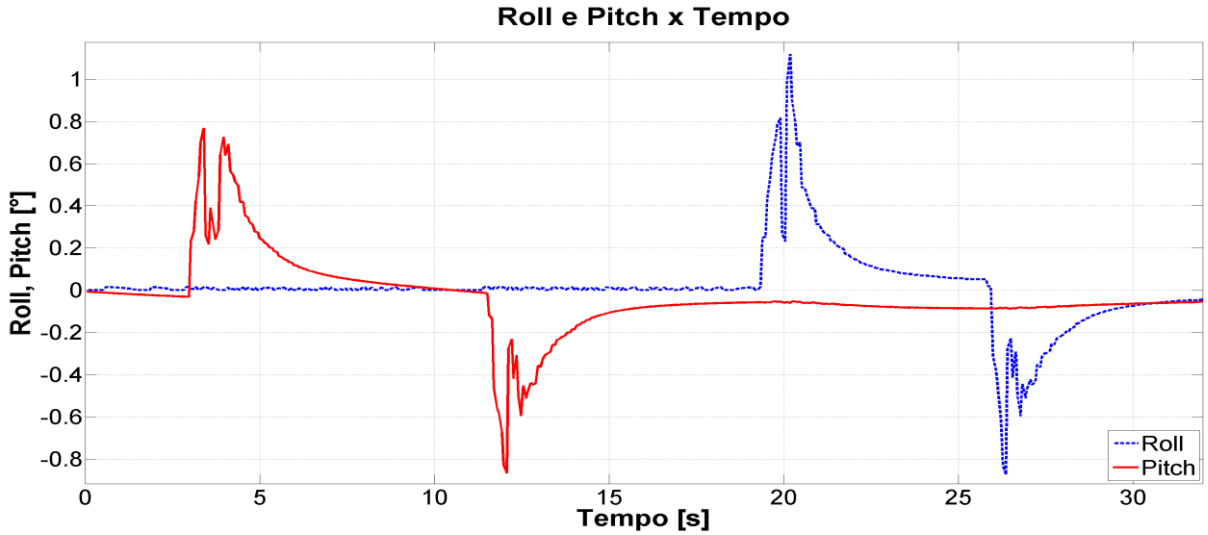


Figura 59 - SIL, MSFS minimizado, Gráfico Roll e Pitch

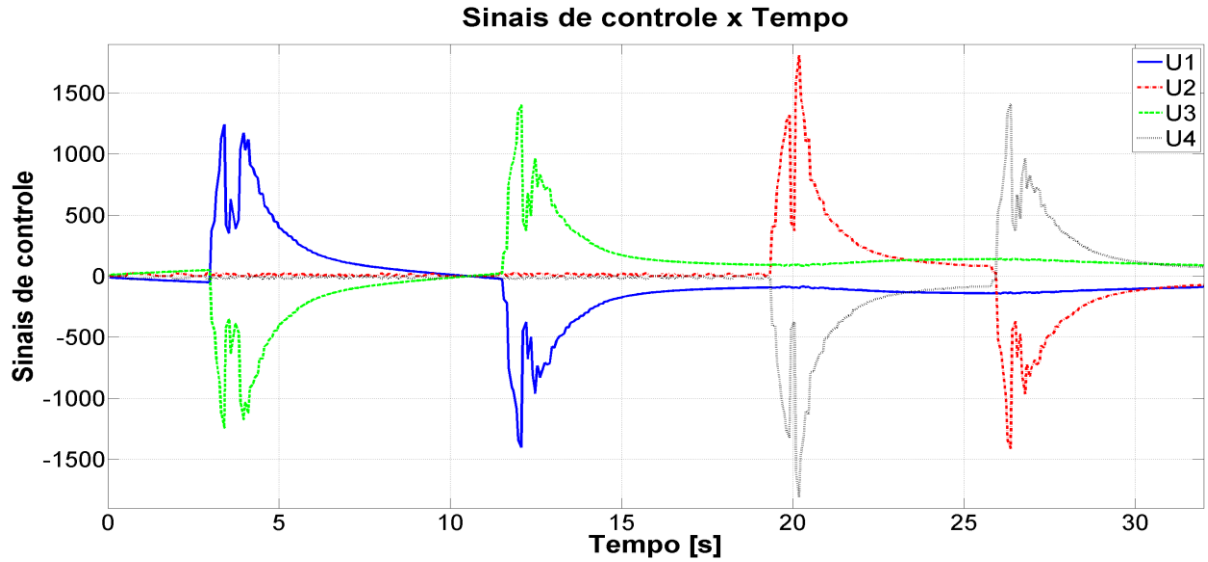


Figura 60 - SIL, MSFS minimizado, Gráfico dos Sinais de Controle

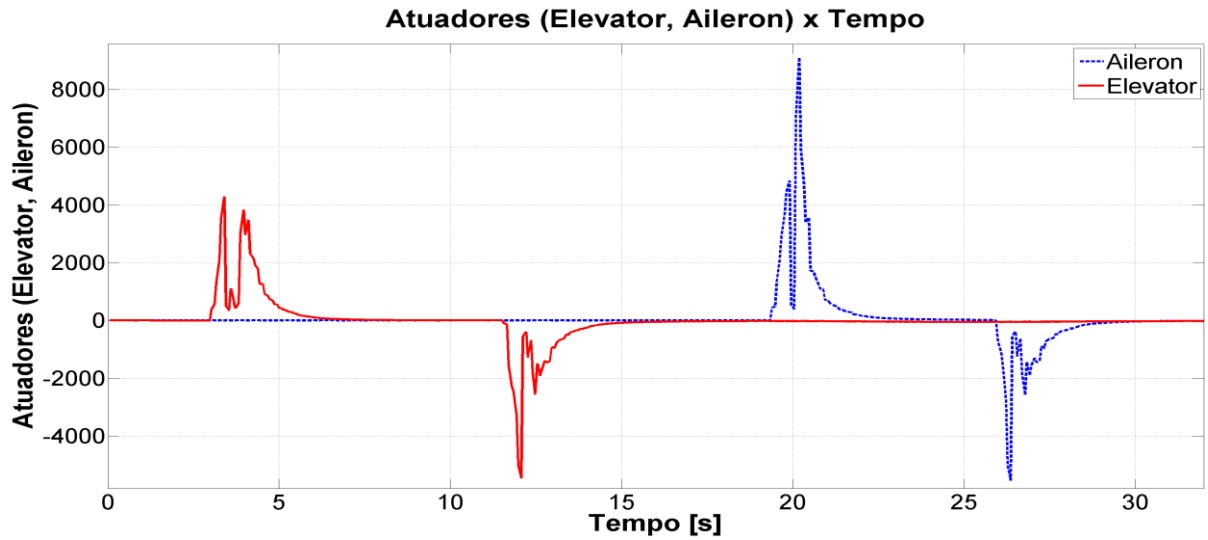


Figura 61 - SIL, MSFS minimizado, Gráfico de Atuadores

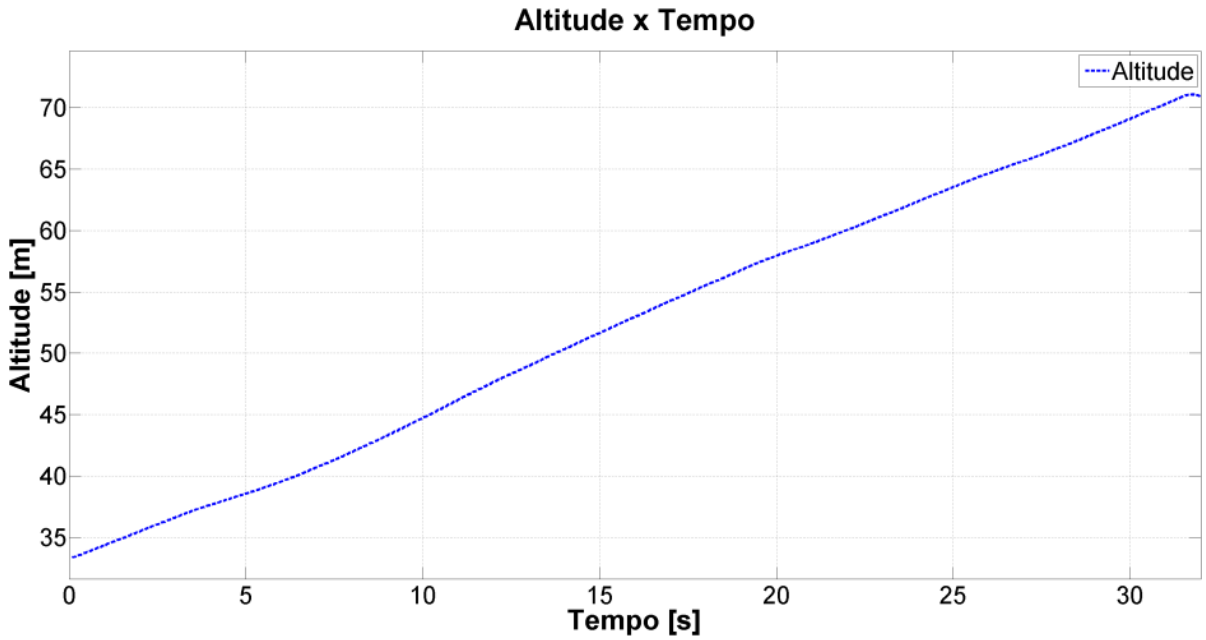


Figura 62 - SIL, MSFS minimizado, Gráfico da Altitude

Os valores mínimo, máximo e médio obtidos para a frequência de execução do controle foram respectivamente 12,82Hz, 21,74Hz e 14,30Hz (Figura 63). Os valores mínimo, máximo e médio obtidos para a período de execução do controle foram respectivamente 46ms, 78ms e 69,92ms (Figura 58).

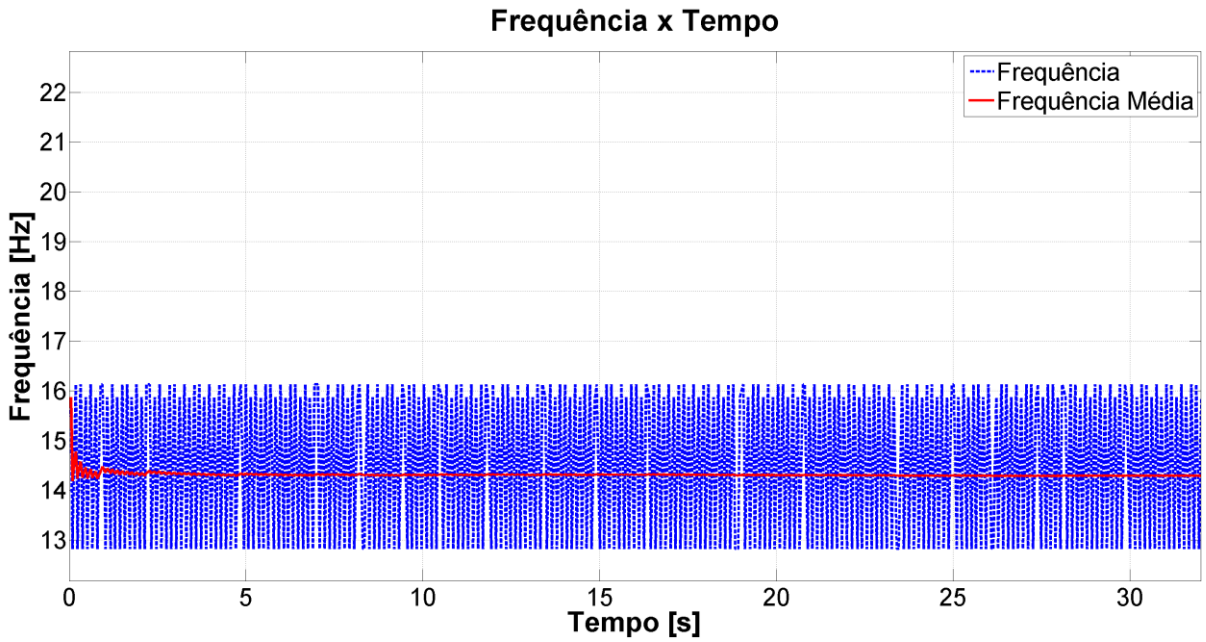


Figura 63 - SIL, MSFS minimizado, Gráfico da Frequência

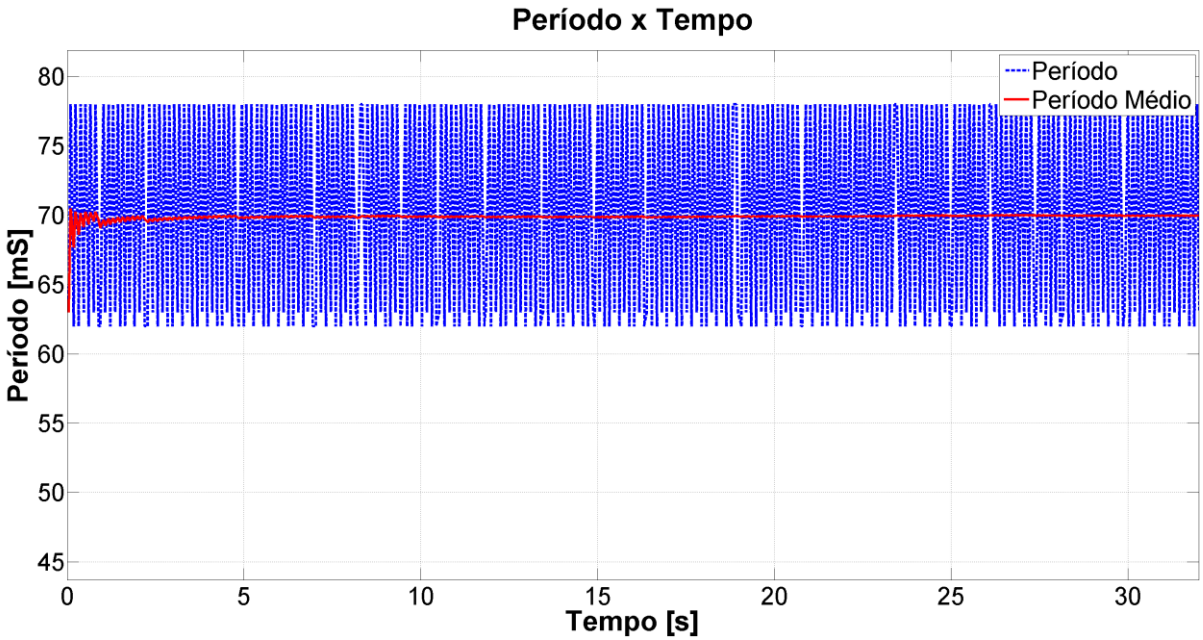


Figura 64 - SIL, MSFS minimizado, Gráfico do Período

5.4.3. Resultados HIL com MSFS maximizado

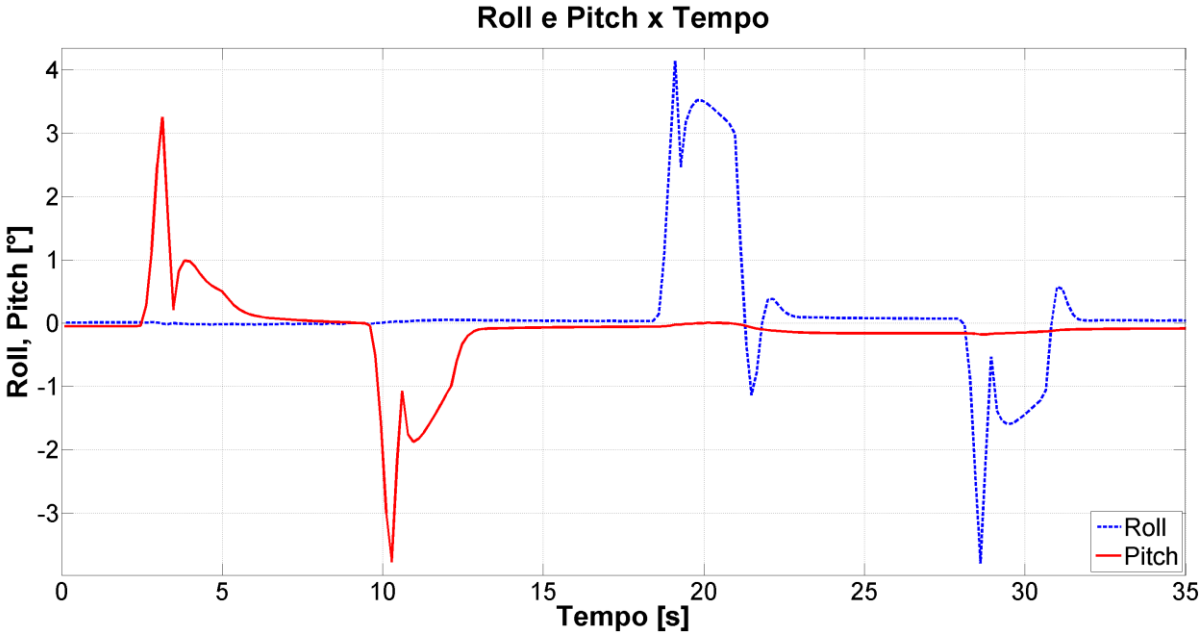


Figura 65 - HIL, MSFS maximizado, Gráfico Roll e Pitch

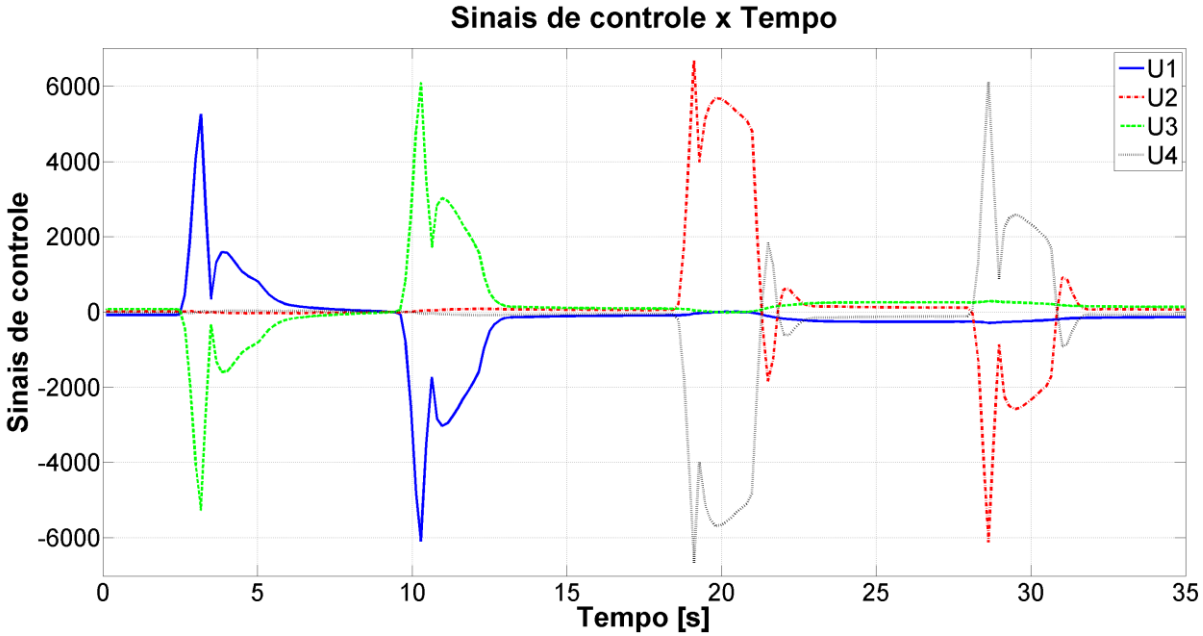


Figura 66 - HIL, MSFS maximizado, Gráfico dos Sinais de Controle

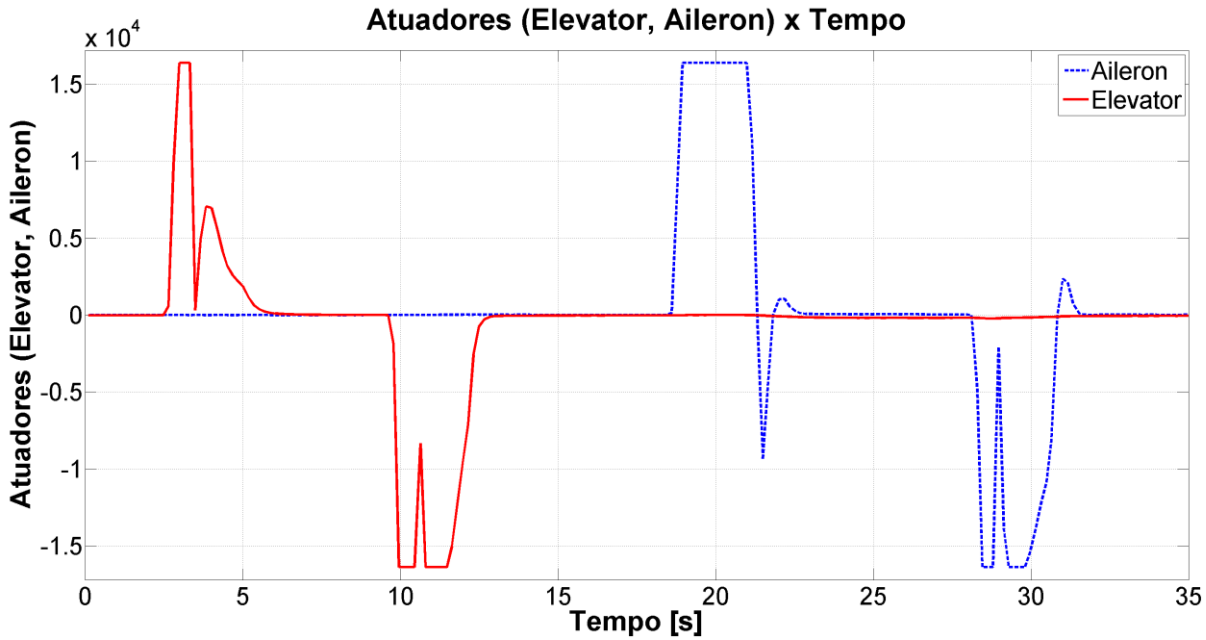


Figura 67 - HIL, MSFS maximizado, Gráfico dos Atuadores

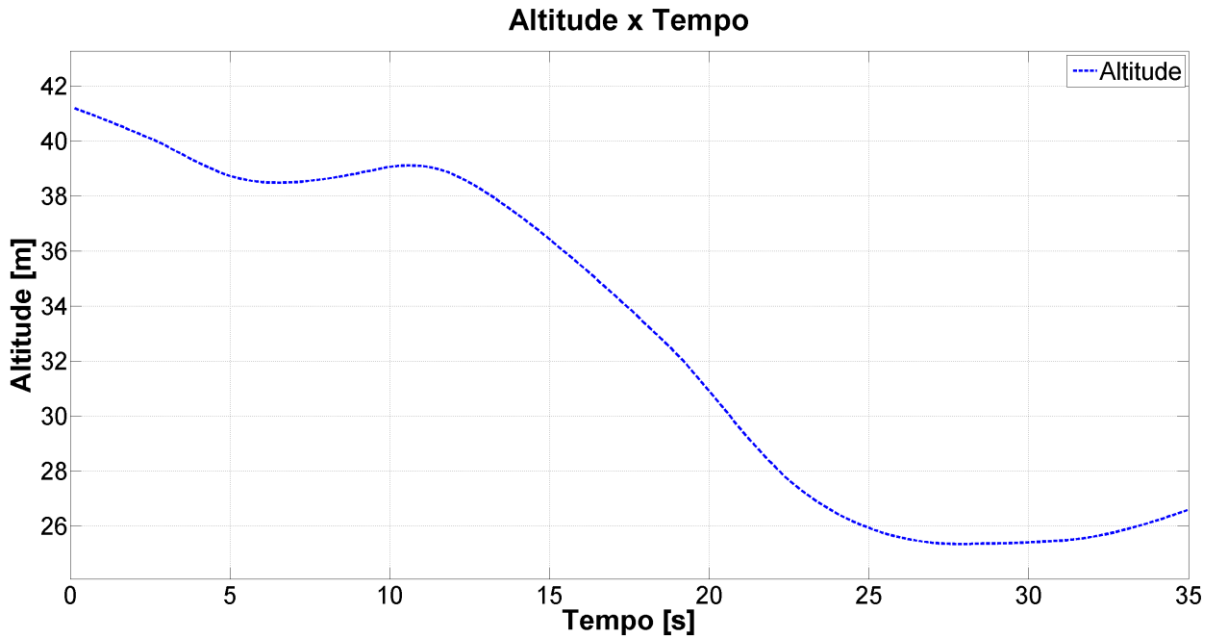


Figura 68 - HIL, MSFS maximizado, Gráfico da altitude

Os valores mínimo, máximo e médio obtidos para a frequência de execução do controle foram respectivamente 5,81Hz, 6,41Hz e 5,89Hz (Figura 63). Os valores mínimo, máximo e médio obtidos para a período de execução do controle foram respectivamente 156ms, 172ms e 169,69ms (Figura 58).

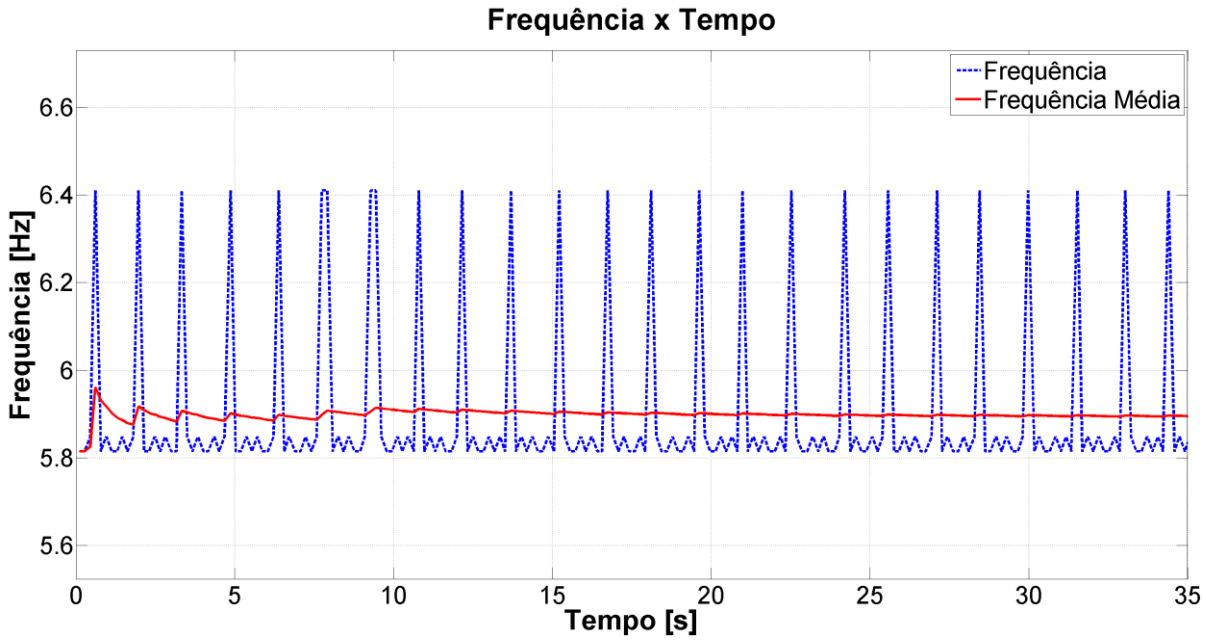


Figura 69 - HIL, MSFS maximizado, Gráfico da frequência

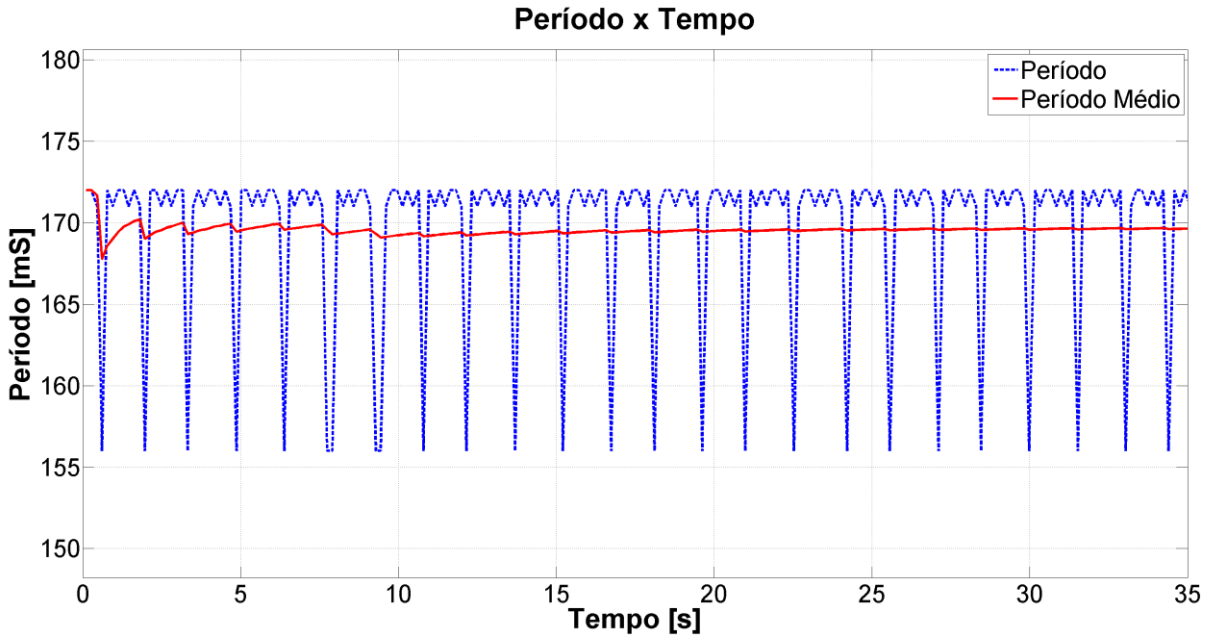


Figura 70 - HIL, MSFS maximizado, Gráfico do período

5.4.4. Resultados HIL com MSFS minimizado

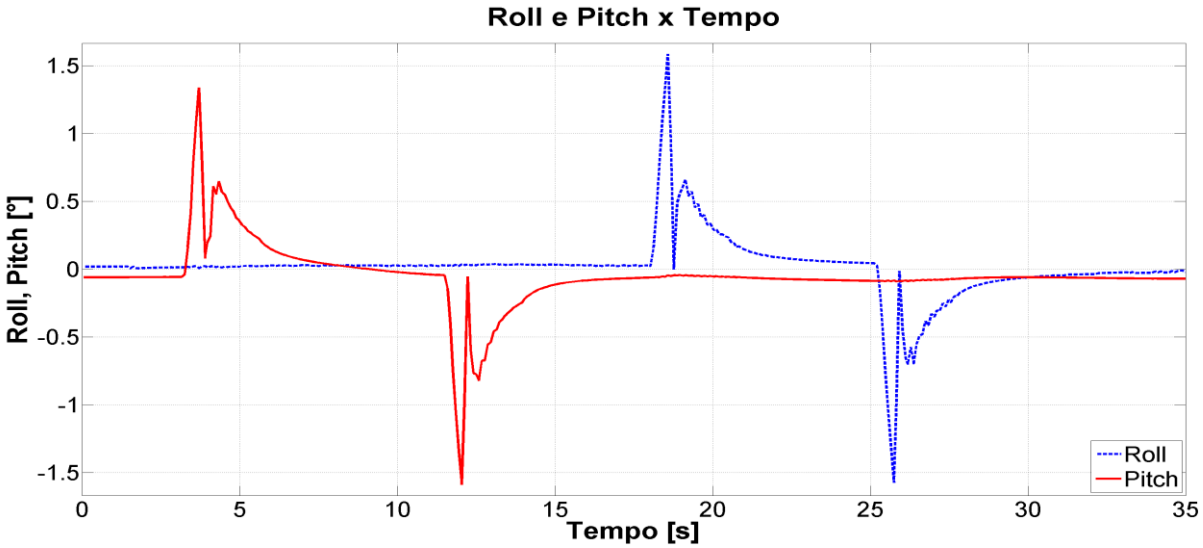


Figura 71 - HIL, MSFS minimizado, Gráfico de Roll e Pitch

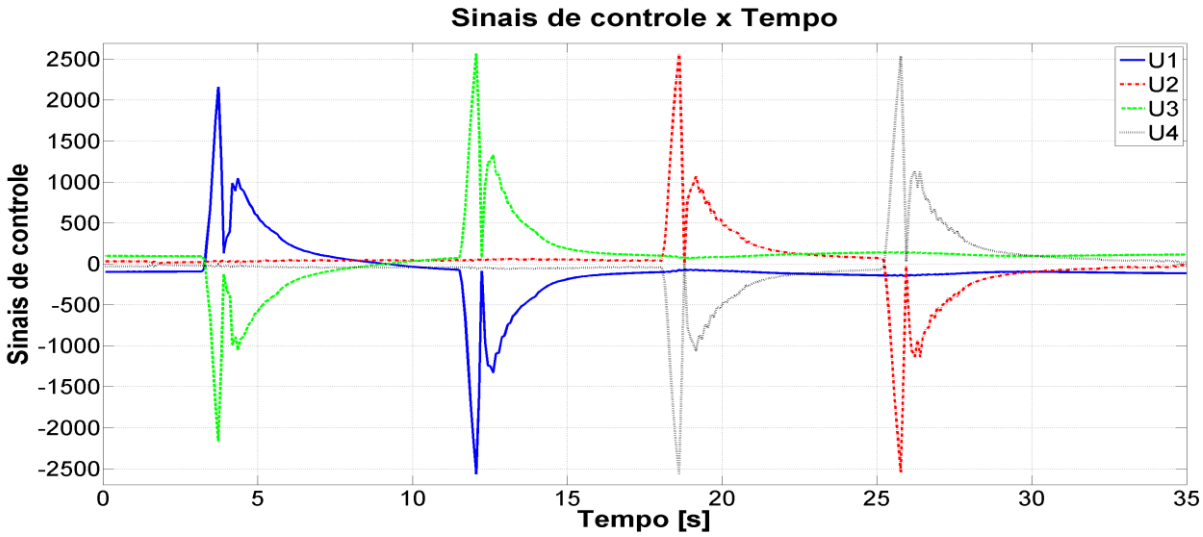


Figura 72 - HIL, MSFS minimizado, Gráfico dos Sinais de Controle

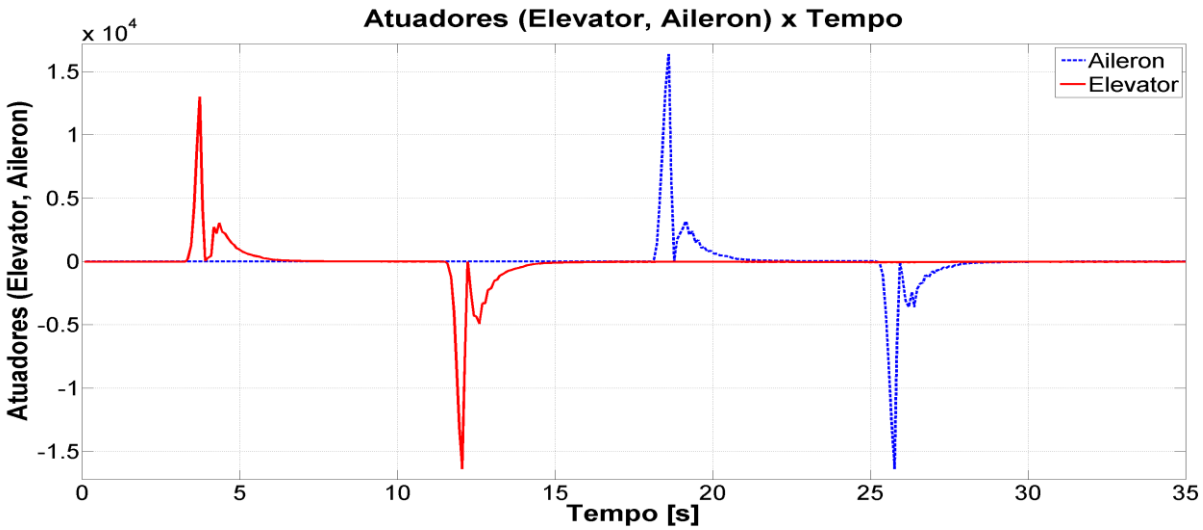


Figura 73 - HIL, MSFS minimizado, Gráfico dos Atuadores

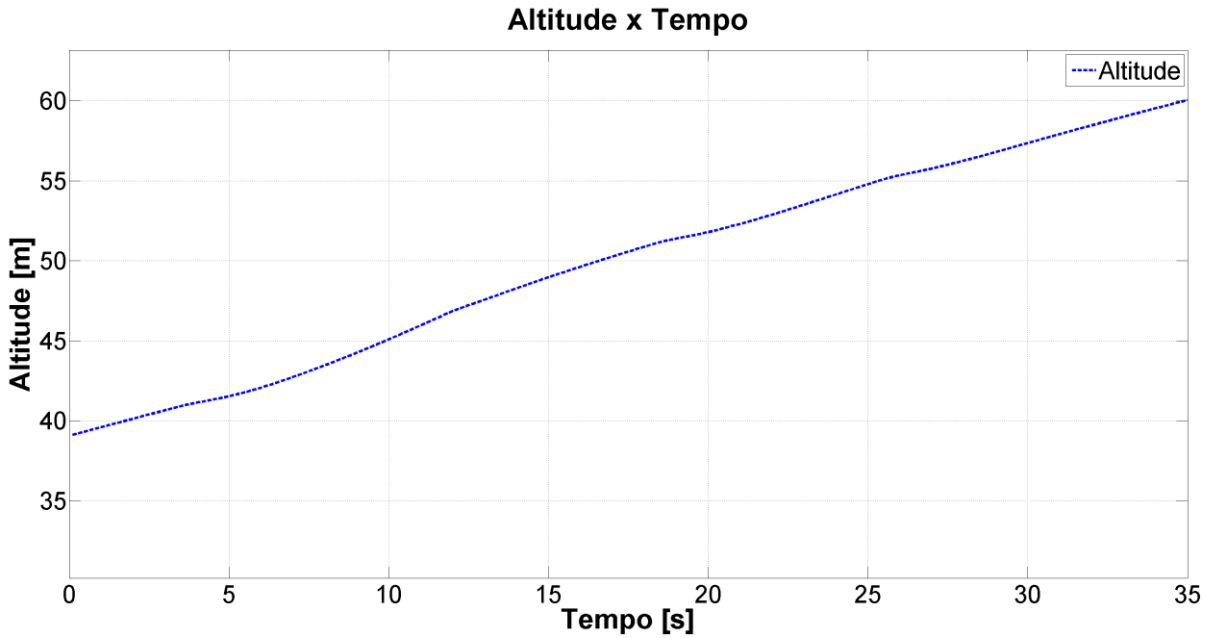


Figura 74 - HIL, MSFS minimizado, Gráfico da Altitude

Os valores mínimo, máximo e médio obtidos para a frequência de execução do controle foram respectivamente 9,09Hz, 12,82Hz e 11,02Hz (Figura 63). Os valores mínimo, máximo e médio obtidos para a período de execução do controle foram respectivamente 78ms, 110ms e 90,71ms (Figura 58).

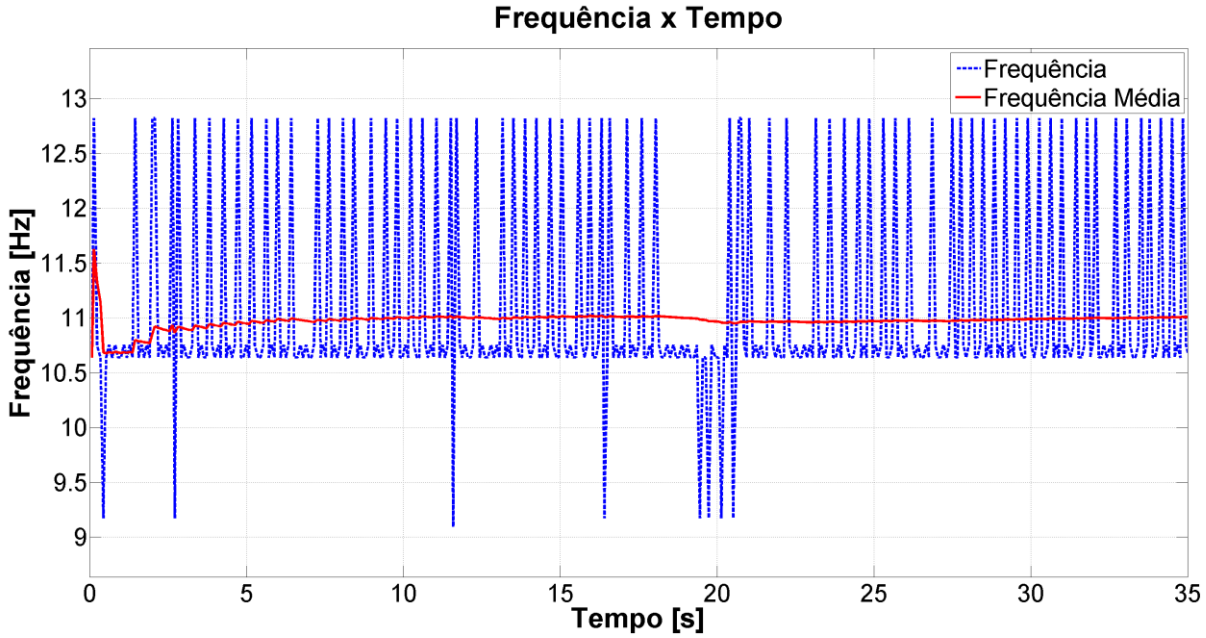


Figura 75 - HIL, MSFS minimizado, Gráfico da frequência

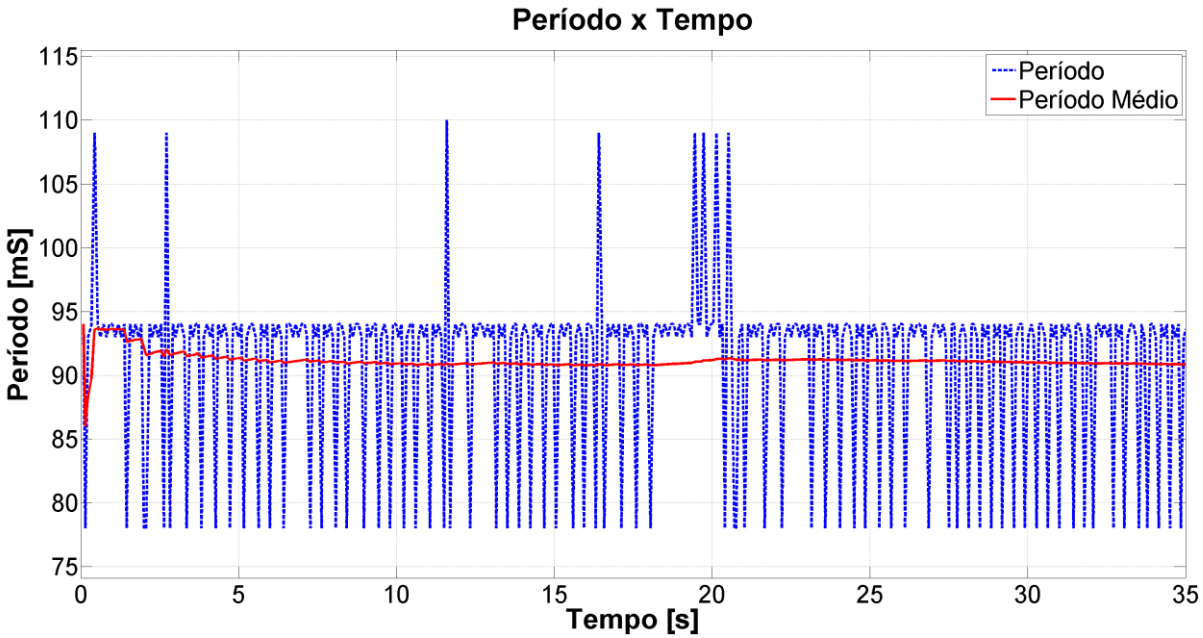


Figura 76 - HIL, MSFS minimizado, Gráfico do período

5.5. Teste de estabilidade durante uma tempestade ligando e desligando o controle.

Nos testes da seção 5.4 e da seção 5.6, o distúrbio gerado no joystick foi aplicado no mesmo atuador utilizado pelo controlador para a correção dos ângulos de atitude. No entanto, seria interessante avaliarmos o desempenho do controlador, e do sistema como um todo, em situações mais reais. A melhor opção encontrada para este propósito foi alterando as condições atmosféricas dentro do MSFS. As mudanças de velocidade do ar e até mesmo a precipitação de chuva ou neve, geram distúrbios no controle que não são aplicados nas superfícies de controle utilizadas pelo controlador.

Nas simulações, no entanto, não foi possível visualizar mudanças significativas dos ângulos de atitude e a respectiva correção do controlador simplesmente colocando o helicóptero em meio à tempestade. A melhor forma de evidenciar isso foi ligando e desligando a execução do controle. Dessa forma, com o controle desligado, a aeronave gradativamente saia da posição 0 e, ao ser ligado novamente, sofria a ação de controle que resultava na correção dos ângulos.

A dinâmica desse liga-desliga do controle pode ser facilmente visualizada nas figuras Figura 78, Figura 79, Figura 81 e Figura 82. Na Figura 78 e na Figura 79 pode-se ver o momento em que o controle estava atuante. Comparando com a figura 76, pode-se ver também que ele foi ligado sempre depois de um período de inatividade que resultava na perda de orientação da aeronave. Isso evidencia a eficácia do controlador.

A Figura 81 e a Figura 82, além de contribuírem para a mesma conclusão do parágrafo anterior, ainda mostram a mudança na frequência de varredura. Quando o controle estava desligado, o período de cada ciclo correspondia ao tempo decorrido no procedimento de leitura das variáveis apenas, o que resultou numa frequência mais alta. Nos momentos em que o controle é ativado, esses valores voltam a ser compatíveis com os valores comprovados nos testes anteriores. Para este teste especificamente, as curvas de período e frequência médios foram descartados por não serem mais de interesse.

5.5.1. Resultados SIL com MSFS maximizado

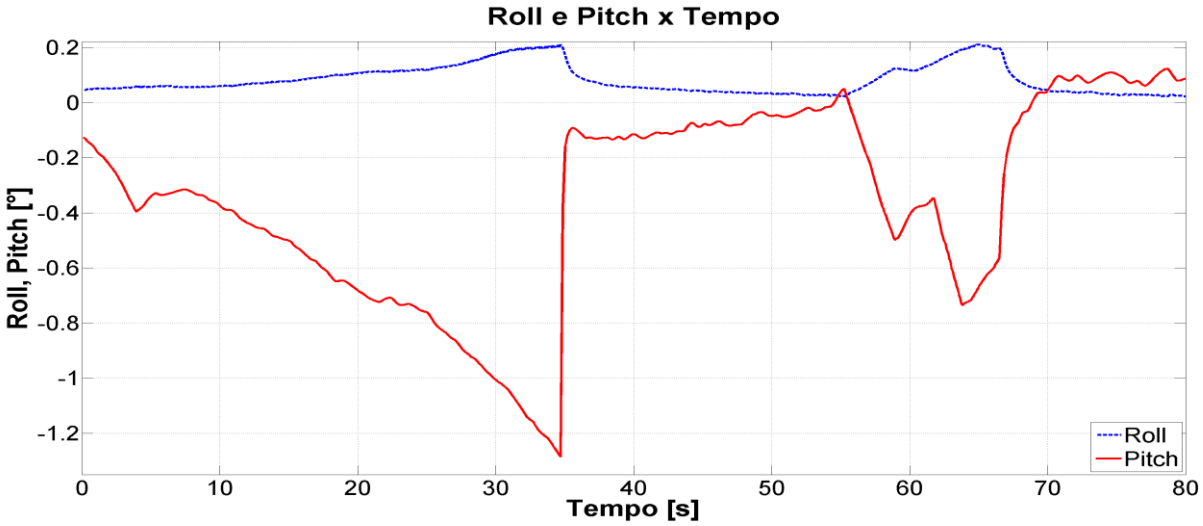


Figura 77 - SIL, MSFS maximizado, Gráfico Roll e Pitch

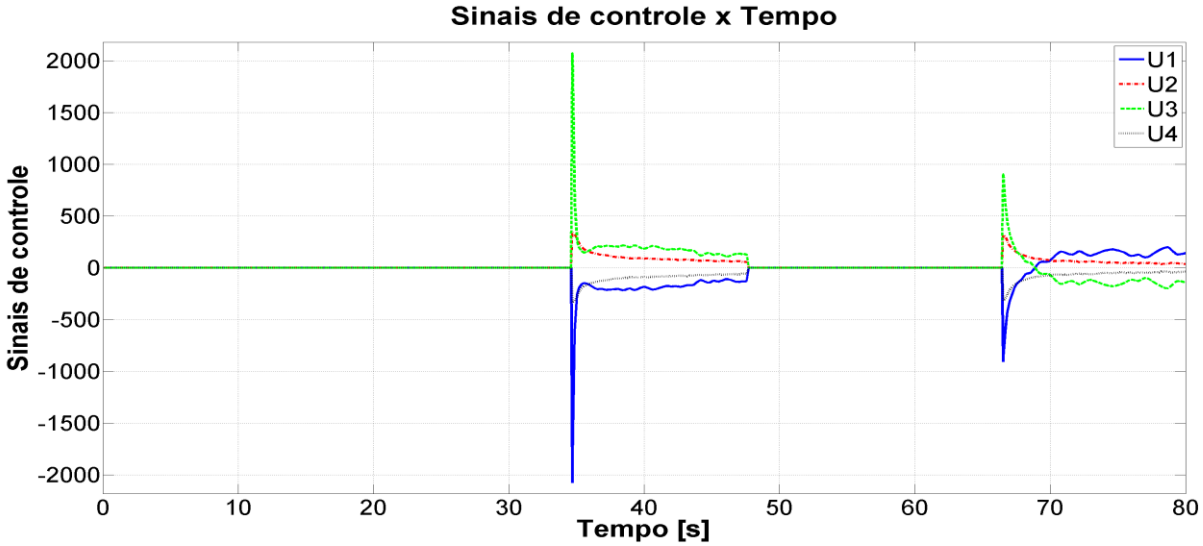


Figura 78 - SIL, MSFS maximizado, Gráfico dos Sinais de Controle

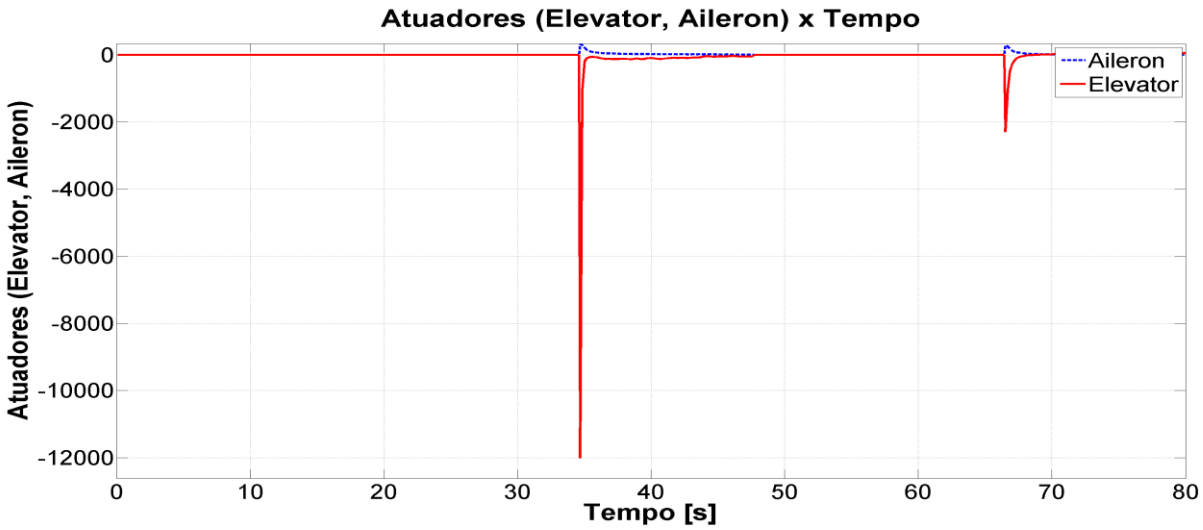


Figura 79 - SIL, MSFS maximizado, Gráfico dos Atuadores

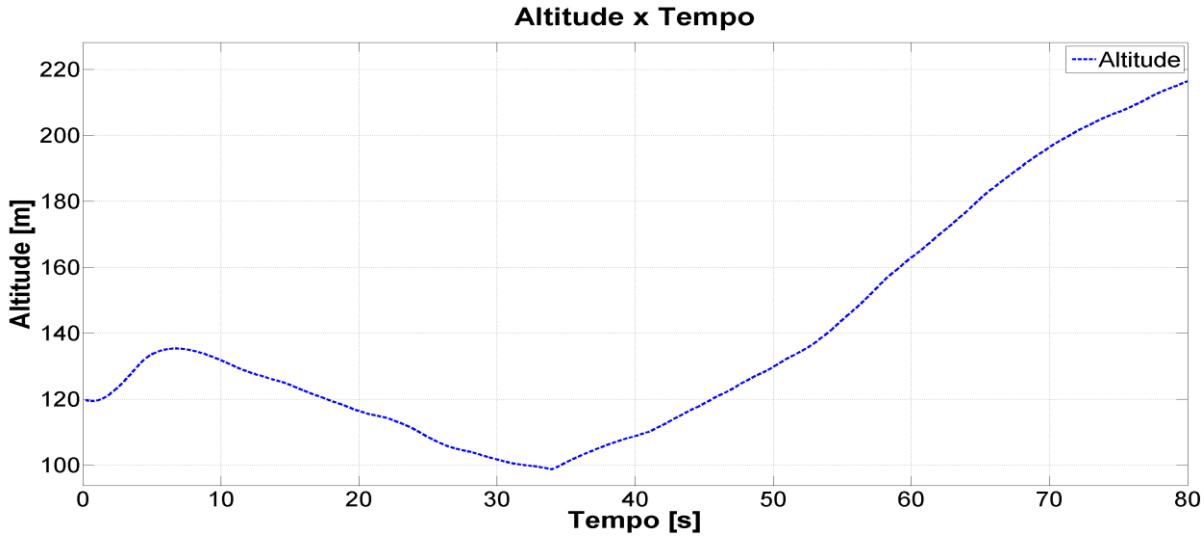


Figura 80 - SIL, MSFS maximizado, Gráfico da Altitude

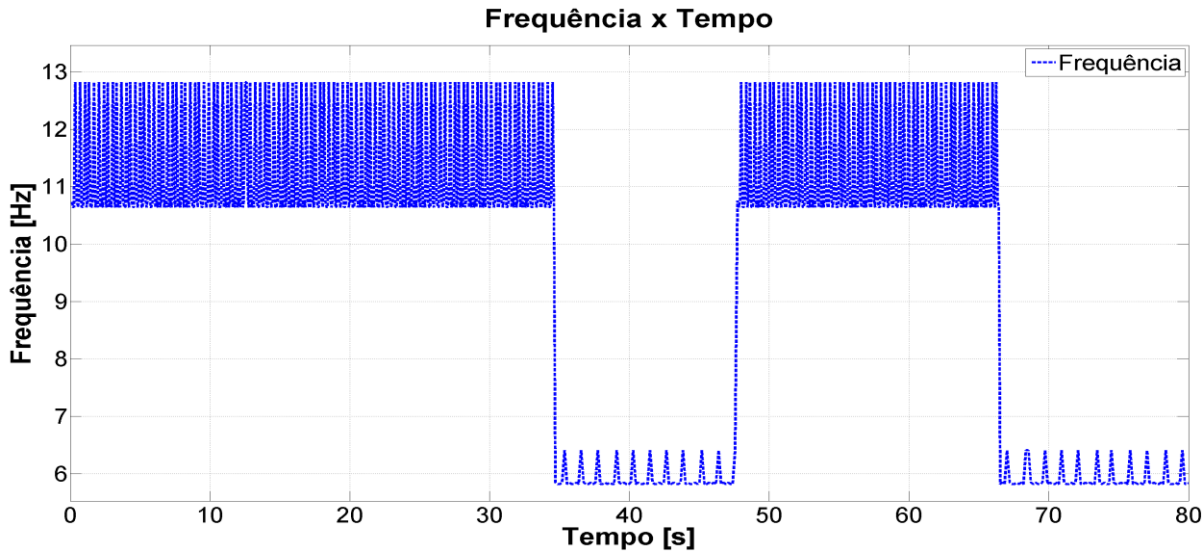


Figura 81 - SIL, MSFS maximizado, Gráfico da Frequência

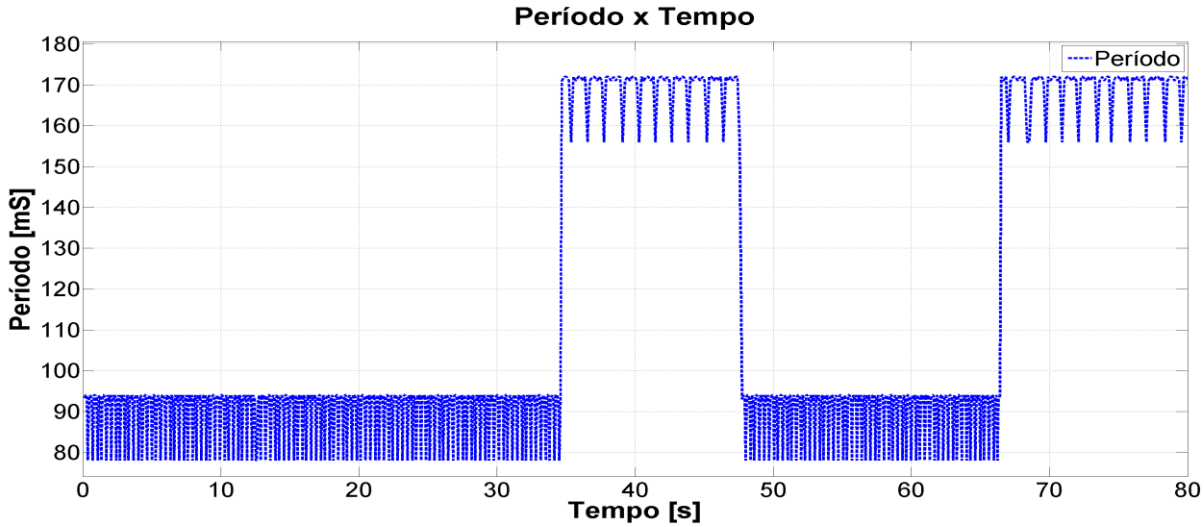


Figura 82 - SIL, MSFS maximizado, Gráfico do Período

5.5.2. Resultados SIL com MSFS minimizado

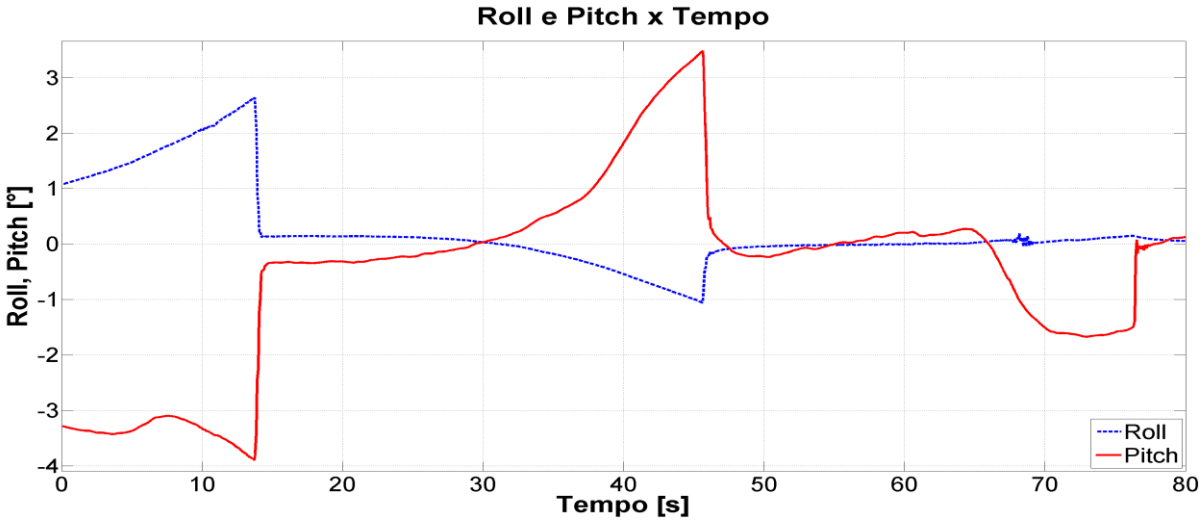


Figura 83 - SIL, MSFS minimizado, Gráfico do Roll e Pitch

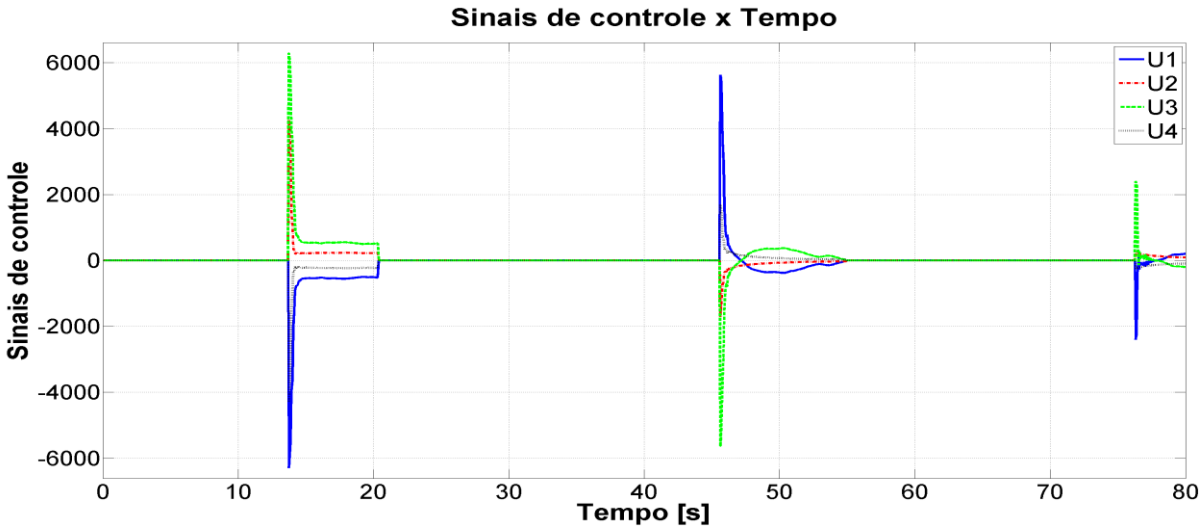


Figura 84 - SIL, MSFS minimizado, Gráfico dos Sinais de Controle

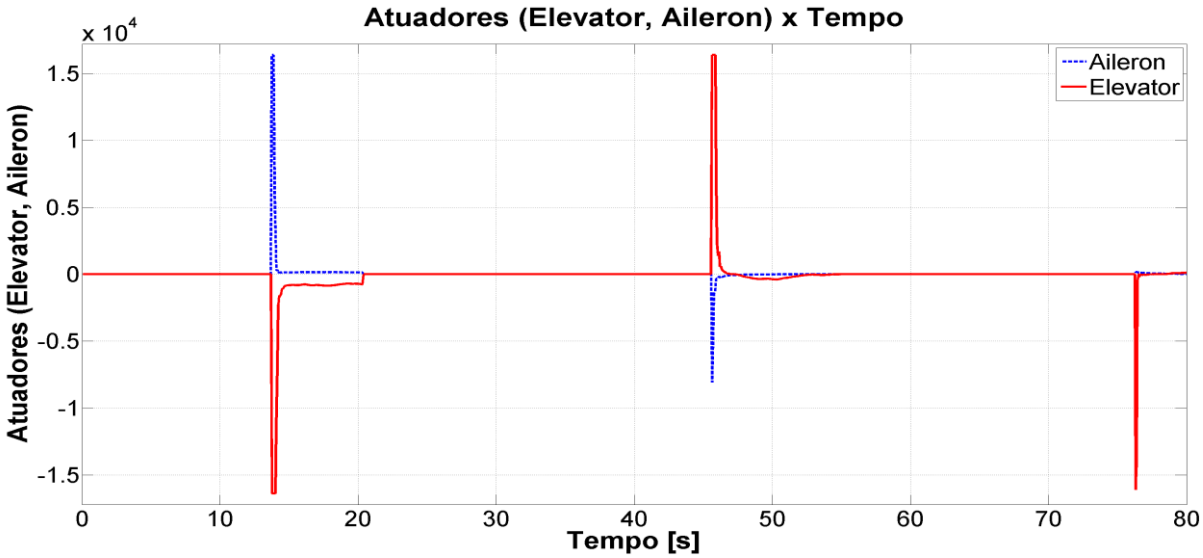


Figura 85 - SIL, MSFS minimizado, Gráfico dos Atuadores

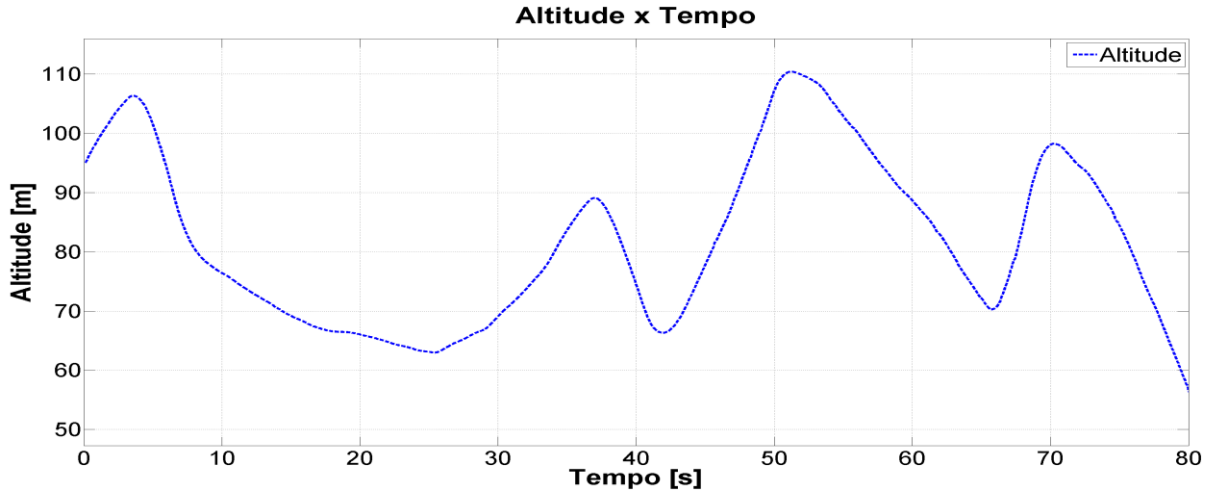


Figura 86 - SIL, MSFS minimizado, Gráfico da Altitude

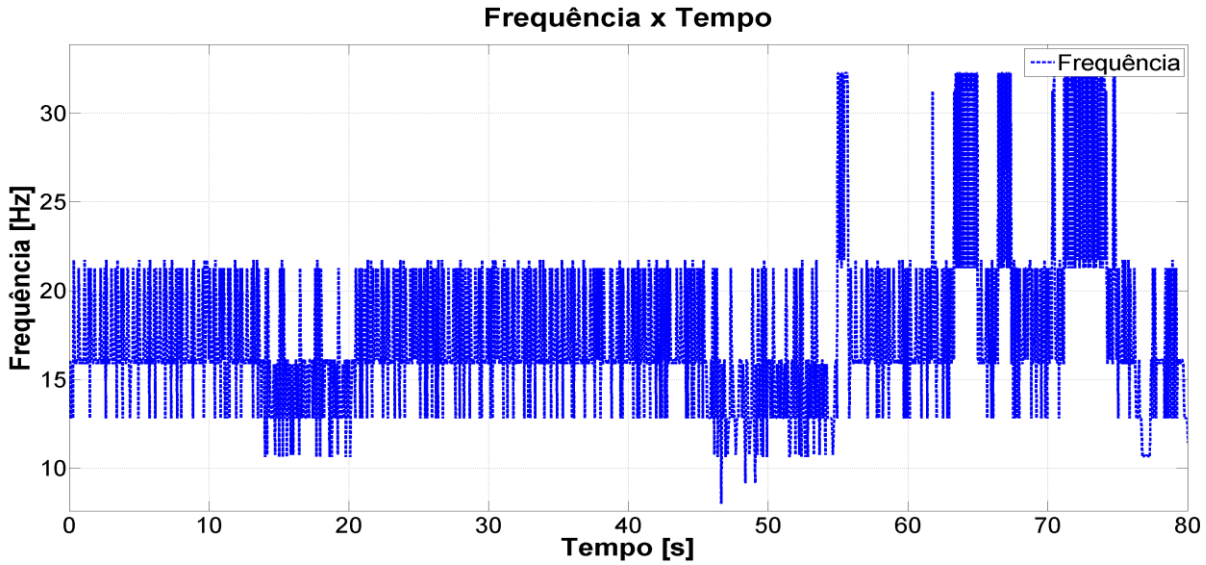


Figura 87 - SIL, MSFS minimizado, Gráfico da Frequência

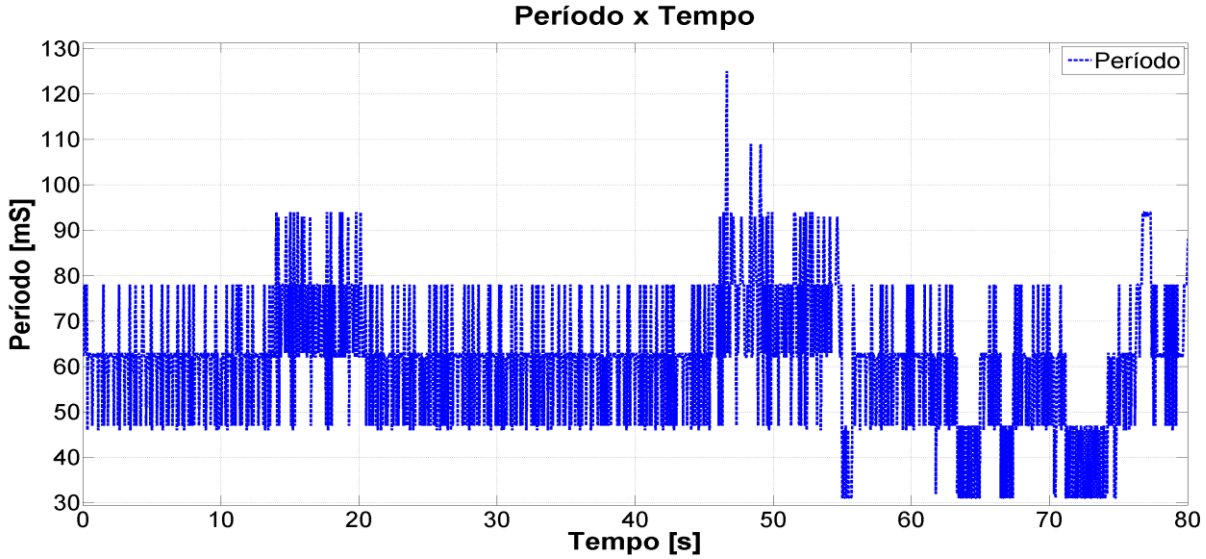


Figura 88 - SIL, MSFS minimizado, Gráfico do Período

5.5.3. Resultados HIL com MSFS maximizado

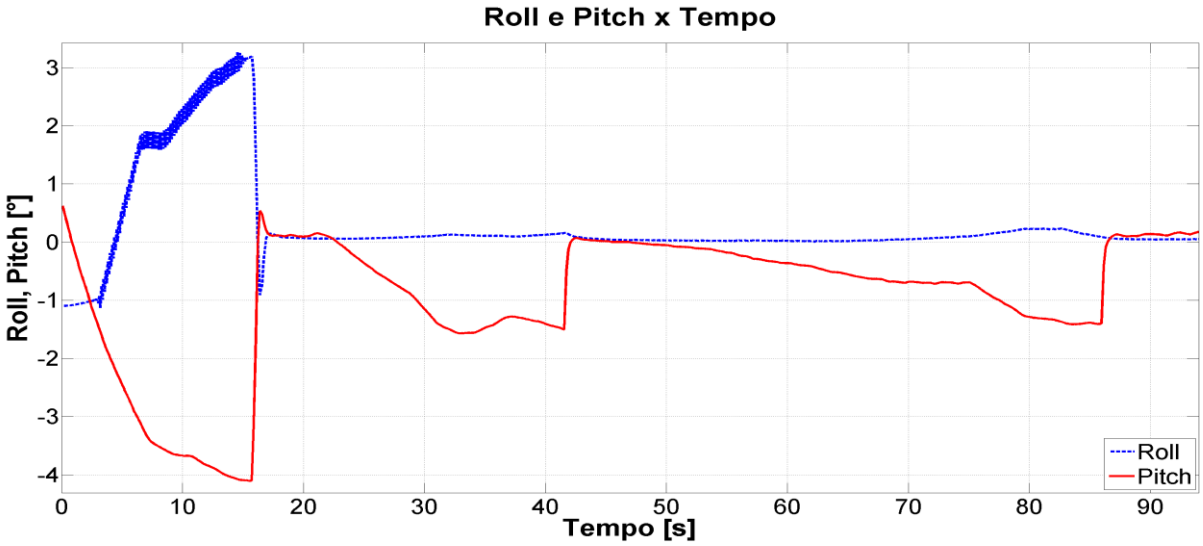


Figura 89 - HIL, MSFS maximizado, Gráfico de Roll e Pitch

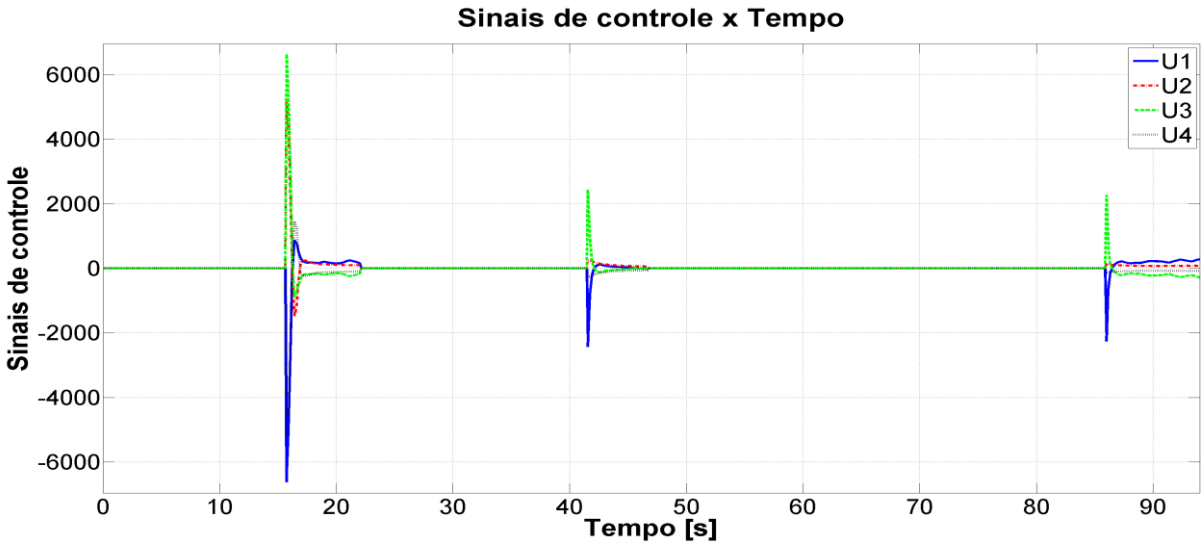


Figura 90 - HIL, MSFS maximizado, Gráfico dos Sinais de Controle

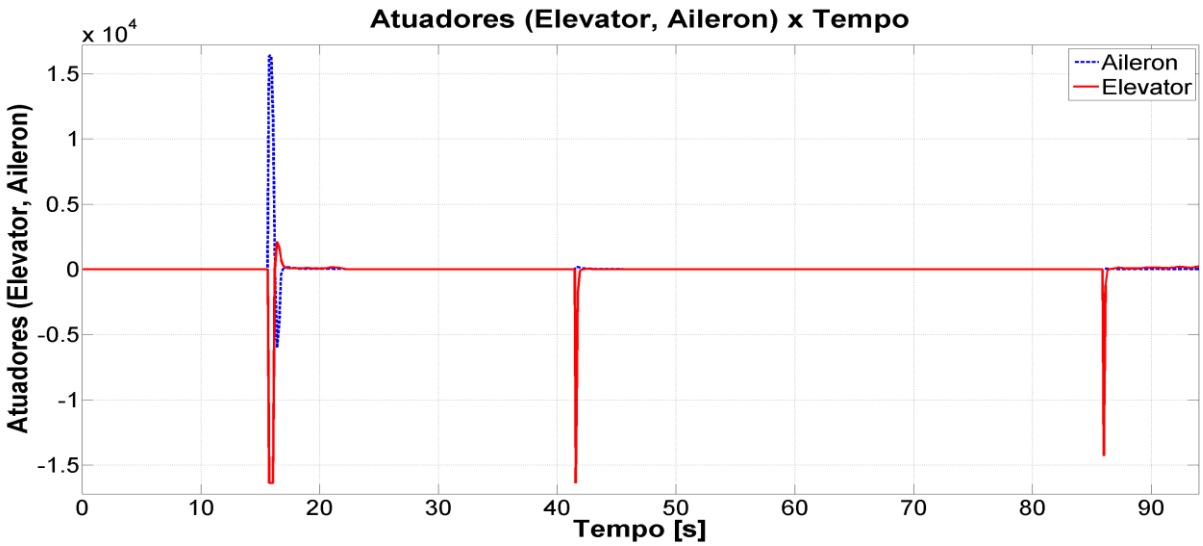


Figura 91 - HIL, MSFS maximizado, Gráfico do Atuadores

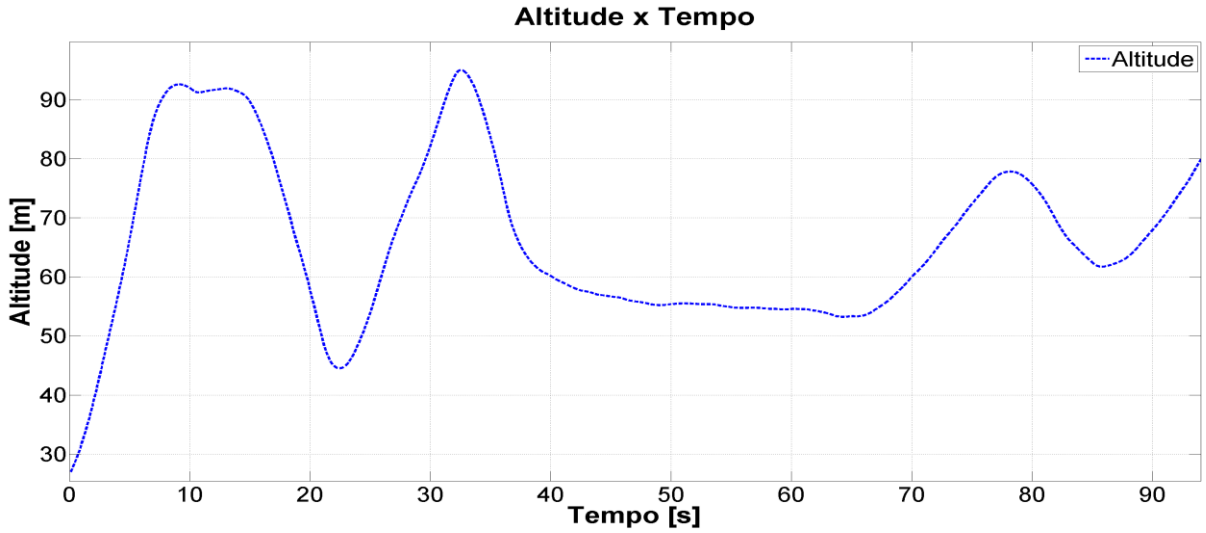


Figura 92 - HIL, MSFS maximizado, Gráfico da Altitude

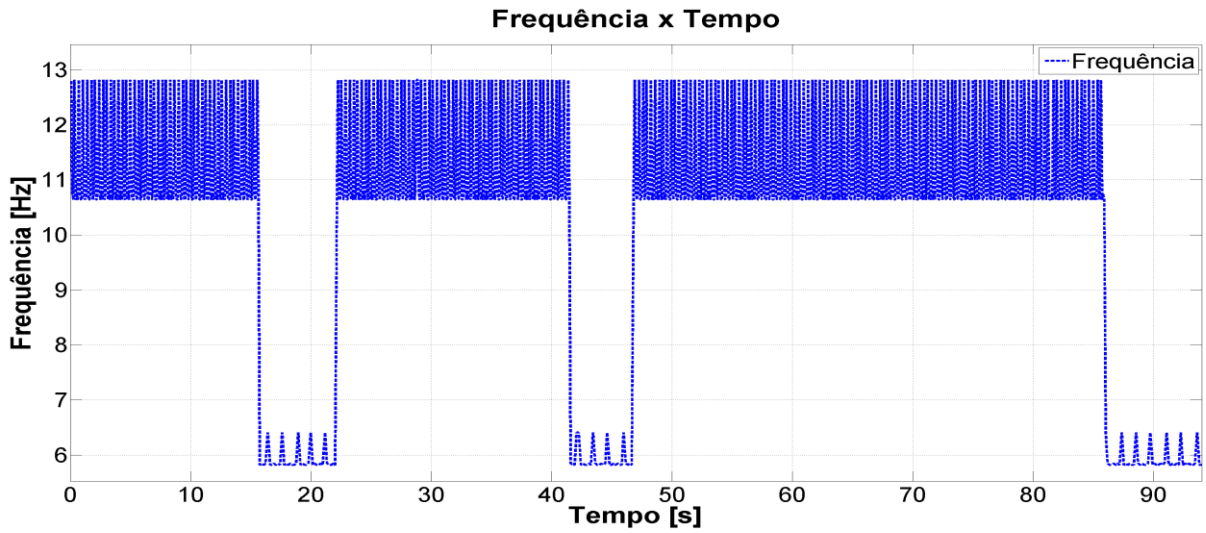


Figura 93 - HIL, MSFS maximizado, Gráfico da Frequência

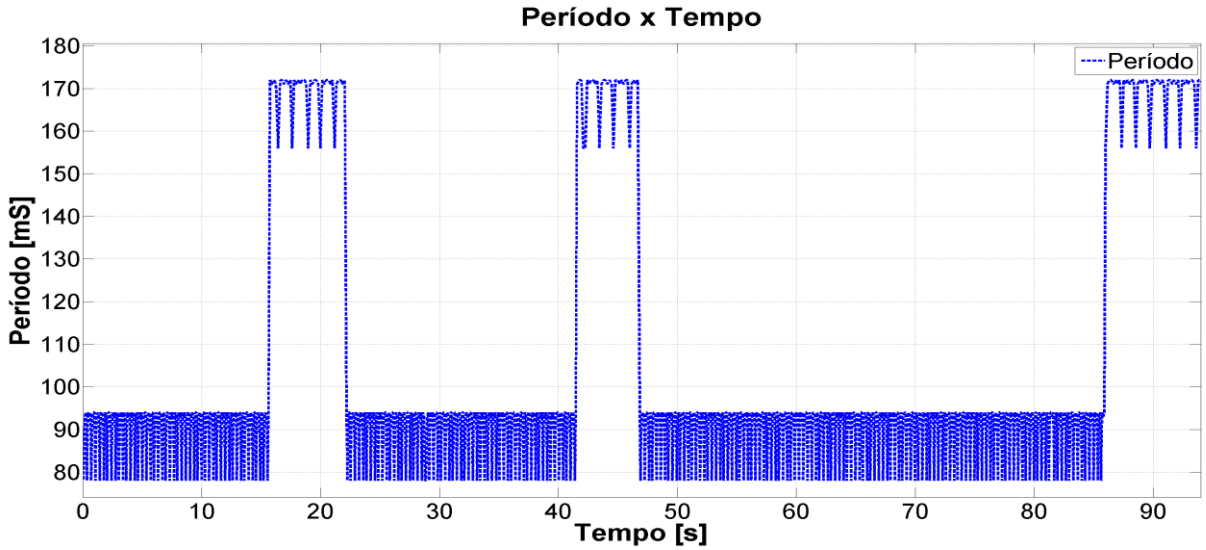


Figura 94 - HIL, MSFS maximizado, Gráfico do Período

5.5.4. Resultados HIL com MSFS minimizado

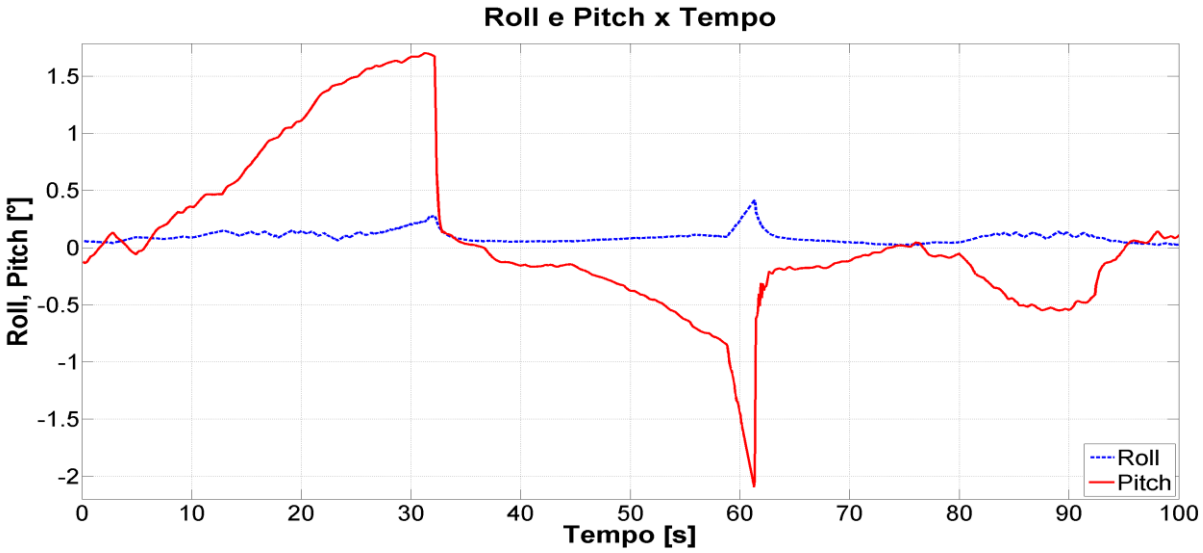


Figura 95 - HIL, MSFS minimizado, Gráfico de Roll e Pitch

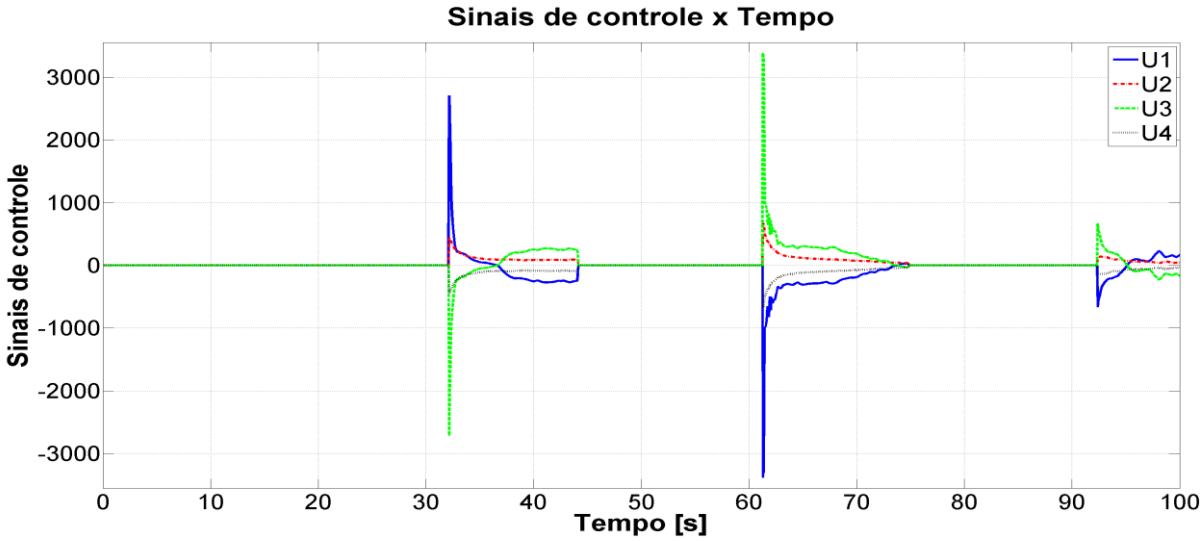


Figura 96 - HIL, MSFS minimizado, Gráfico dos Sinais de Controle

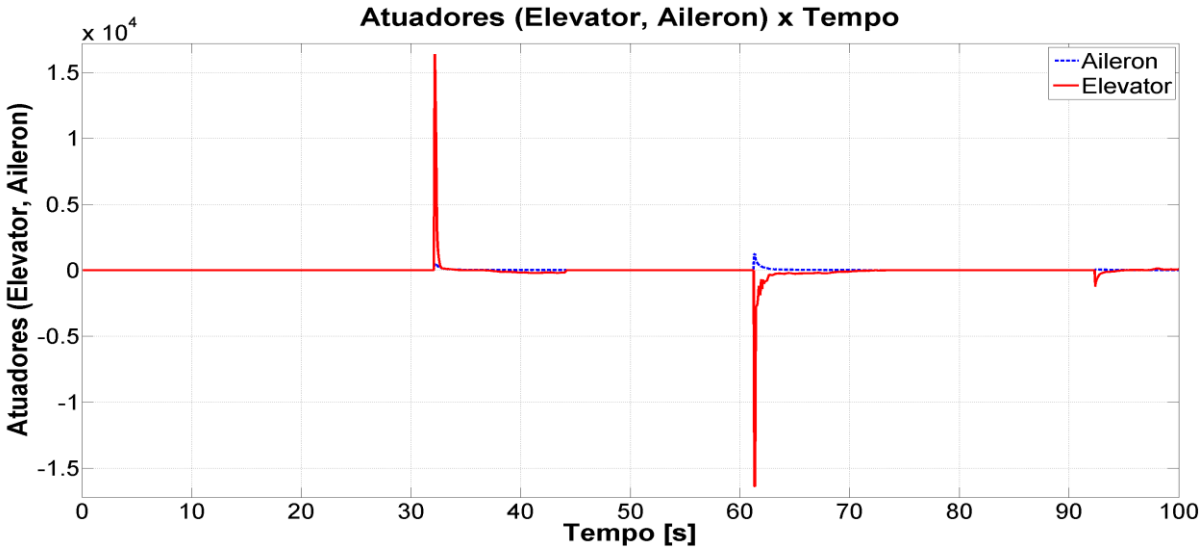


Figura 97 - HIL, MSFS minimizado, Gráfico dos Atuadores

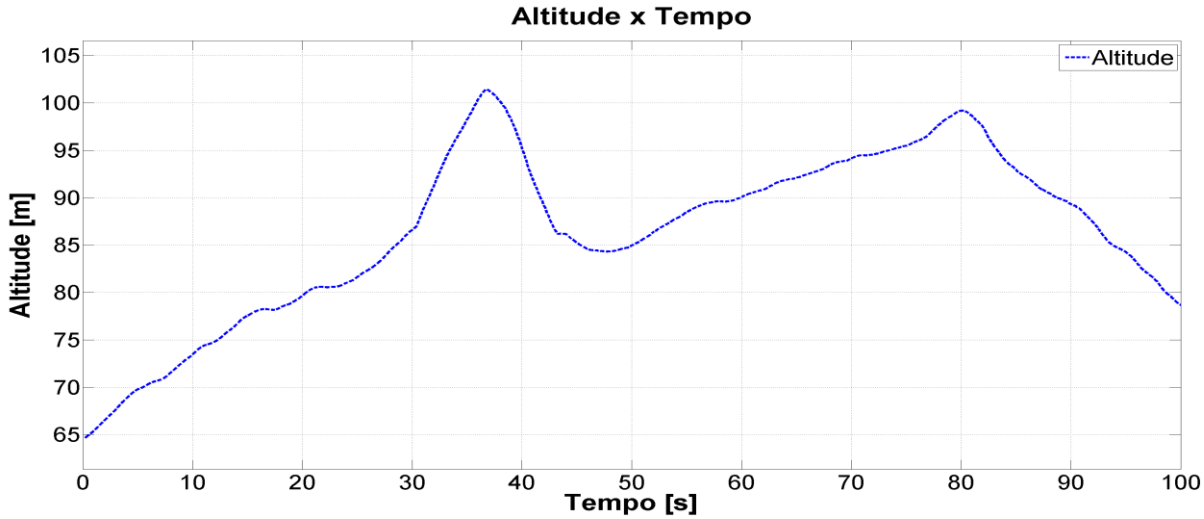


Figura 98 - HIL, MSFS minimizado, Gráfico da Altitude

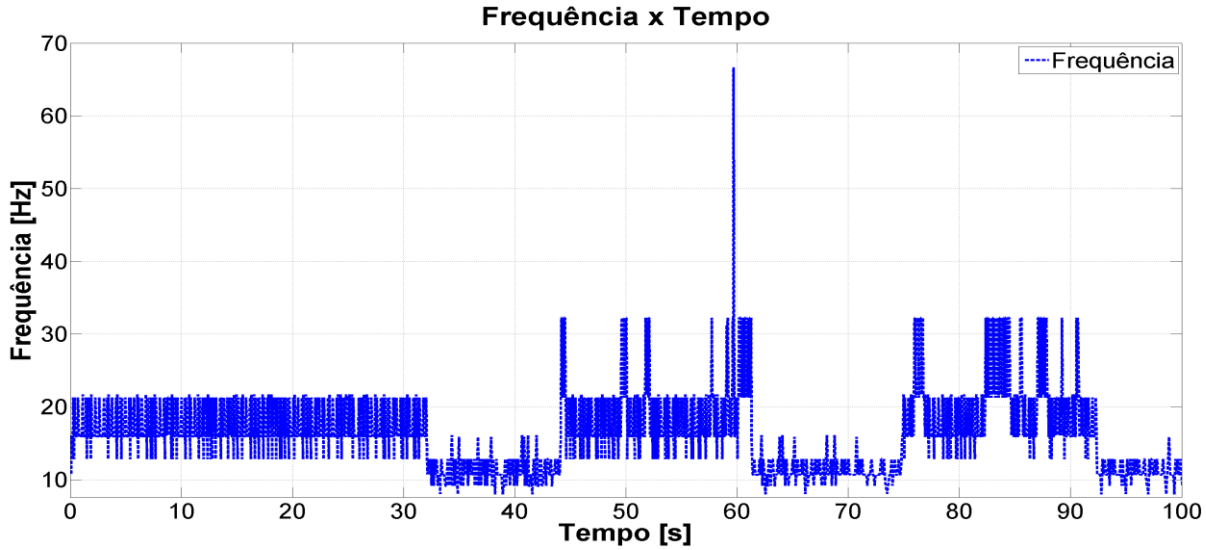


Figura 99 - HIL, MSFS minimizado, Gráfico da frequência

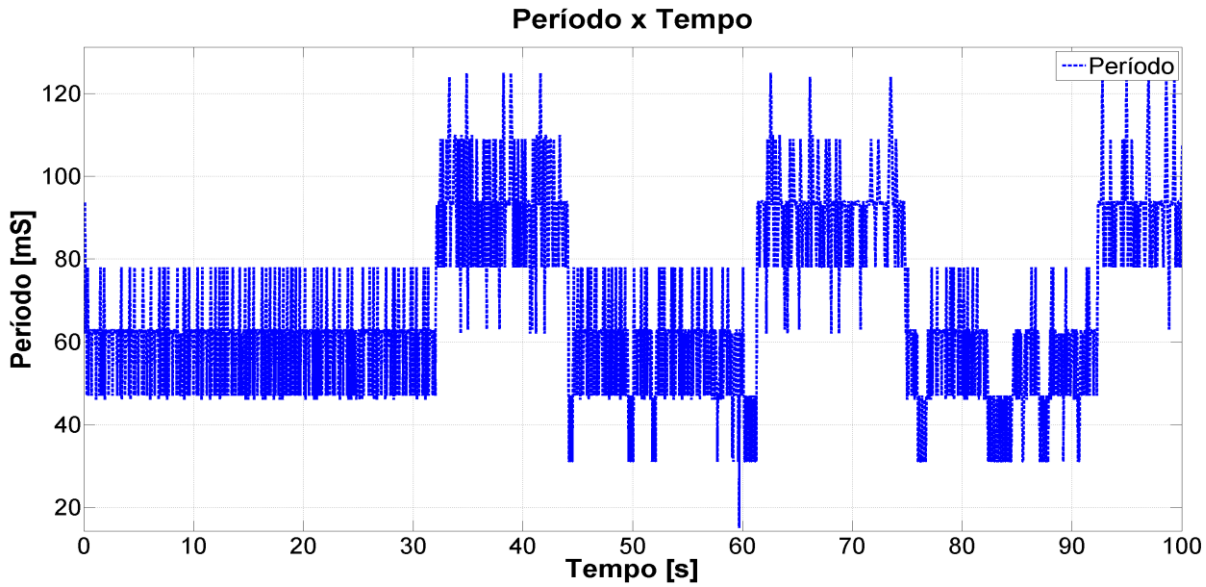


Figura 100 - HIL, MSFS minimizado, Gráfico do Período

5.6. Teste de estabilidade aplicando distúrbio sistemático forçado gerado pelo próprio FVMS

A intenção desse teste é a de avaliar o desempenho do controlador e do sistema como um todo quando um distúrbio sistemático é aplicado nas superfícies de controle da aeronave. Para tanto, foi implementado no próprio FVMS uma rotina, em paralelo com a rotina de execução do controle, responsável por forçar certos valores dos atuadores. O valor máximo que se pode gravar nessas variáveis no MSFS é 16000. Para os testes executados aqui, optou-se por utilizar 12000 no distúrbio.

A frequência utilizada na execução do distúrbio foi de 10Hz em todos os casos, porém, como o sistema operacional, o DELPHI® e o MSFS não são sistemas de tempo real, a frequência real ficou abaixo disso. A Tabela 20 resume os resultados. A média aritmética dos períodos médios e das respectivas frequências de atuação do distúrbio nos casos onde a tela estava maximizado foram de 209,49ms e 4,78Hz. Nos casos em que a tela estava minimizada, foram 173,15ms e 5,83Hz.

Tabela 20 - Período e frequência da varredura de controle e do distúrbio

		Controle		Distúrbio	
Arquitetura	**Tela**	$T_{med}(ms)$	$f_{med}(Hz)$	$T_{med}(ms)$	$f_{med}(Hz)$
SIL	Max	207,78	4,81	209,73	4,77
	Min	71,98	13,89	157,51	6,35
HIL	Max	207,52	4,82	209,24	4,78
	Min	107,89	9,27	188,79	5,3

A Tabela 21 mescla algumas informações da Tabela 20 com a Tabela 17. Ela foi elaborada para que os resultados comparativos fossem mais facilmente visualizados. Como pode ser visto, os valores do período nos testes com a tela maximizada sofreram grandes alterações depois que o distúrbio foi inserido no sistema (aproximadamente 22,5%). No caso do teste com a tela minimizada na arquitetura SIL, a mudança foi bem pequena (3,1%) e na arquitetura HIL um pouco maior (18,97%).

Ainda sobre a Tabela 20, pode-se ver que nos dois testes com a tela maximizada o resultado praticamente acompanhou o resultado do loop de controle. Nos outros dois com a tela minimizada, o que impactou no período maior foi o período de 100ms (10Hz)

estabelecido no *software* FVMS, ou seja, se o limite de estouro do *timer* não estivesse em 100ms, provavelmente a frequência de atuação do distúrbio estaria semelhante às do loop de controle.

Tabela 21 - Tabela comparativa com os valores do período e da frequência entre os testes sem e com a presença do distúrbio sistemático

		Controle Médio (Tabela 17)		Controle Médio (Tabela 20)		Diferença entre os períodos	Porcentagem de aumento no período
Arq.	**Tela**	T_{med} $[ms]$	f_{med} $[Hz]$	T_{med} $[ms]$	f_{med} $[Hz]$	T_{Dif} $[ms]$	$T_{\%}$
SIL	Max	169,67	5,89	207,78	4,81	38,12	22,46
	Min	69,81	14,33	71,98	13,89	2,17	3,10
HIL	Max	169,72	5,89	207,52	4,82	37,80	22,27
	Min	90,69	11,03	107,89	9,27	17,20	18,97

5.6.1. Resultados SIL com MSFS maximizado

A Figura 101 mostra o gráfico de variação dos ângulos de atitude da aeronave durante a aplicação do distúrbio. Note que o valor do ângulo de rolagem, à medida em que o distúrbio era aplicado, passou de -20º. Se o distúrbio não tivesse sido interrompido, a aeronave perderia a estabilidade. A Figura 103 mostra a saturação nos atuadores (16383).

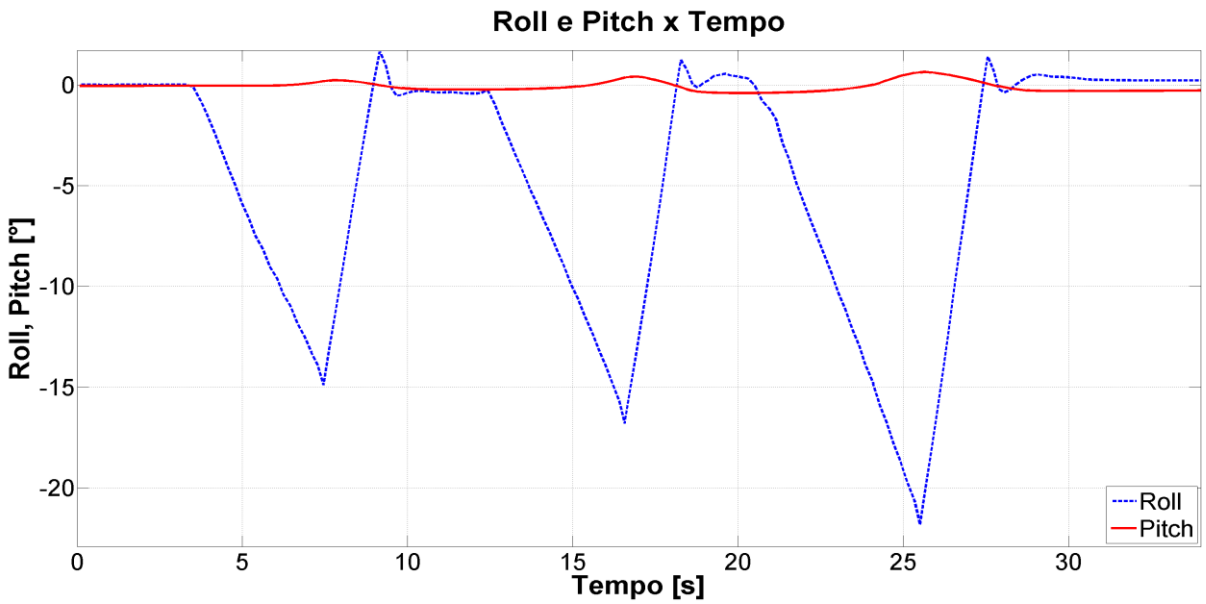


Figura 101 - SIL, MSFS maximizado, Gráfico de Roll e Pitch

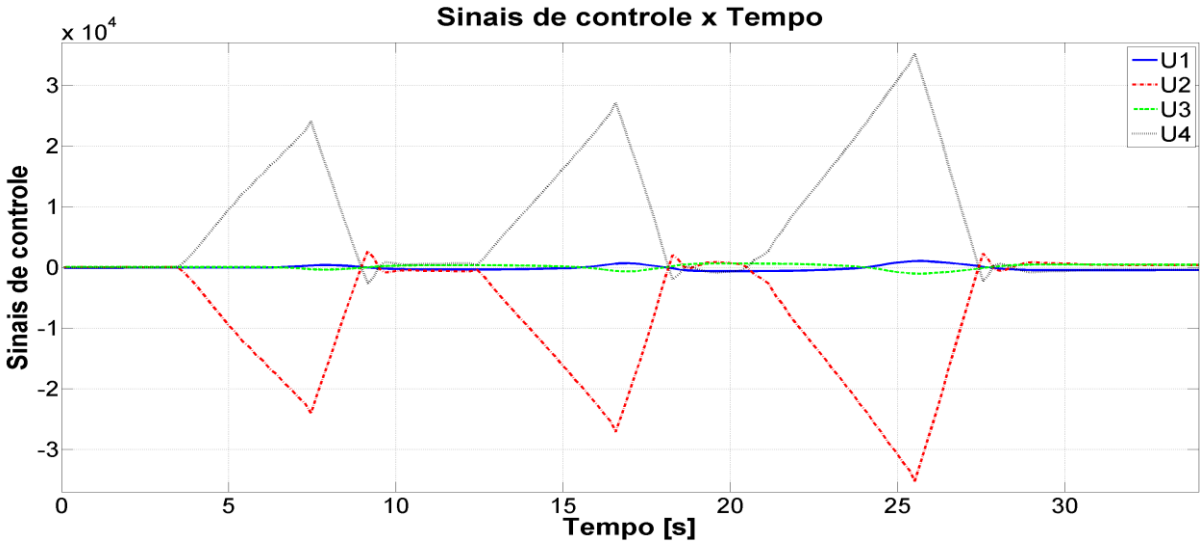


Figura 102 - SIL, MSFS maximizado, Gráfico dos Sinais de Controle

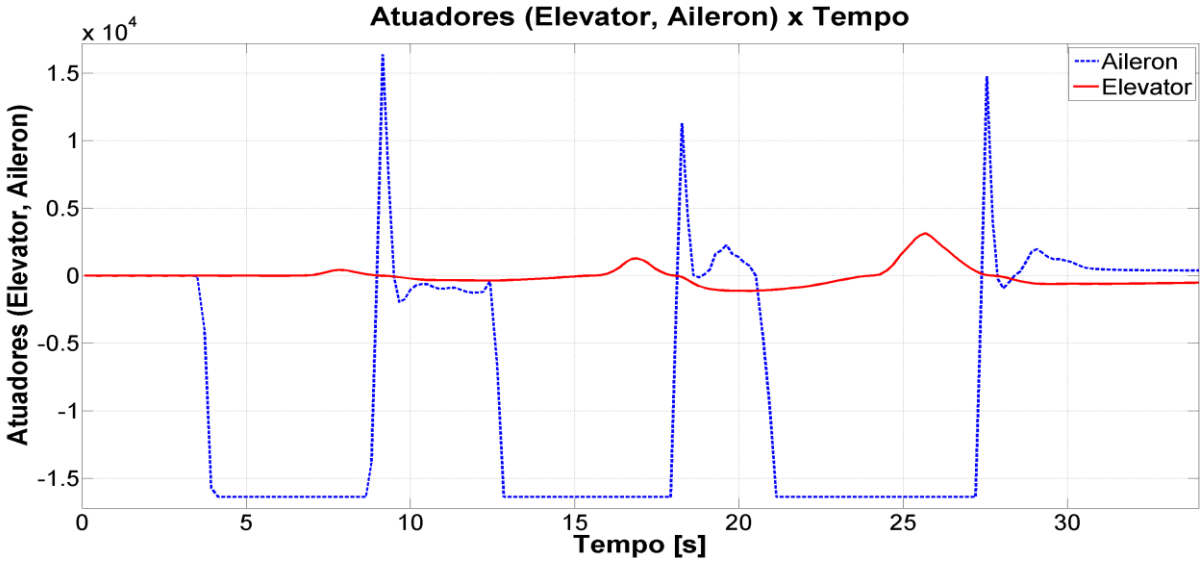


Figura 103 - SIL, MSFS maximizado, Gráfico dos Atuadores

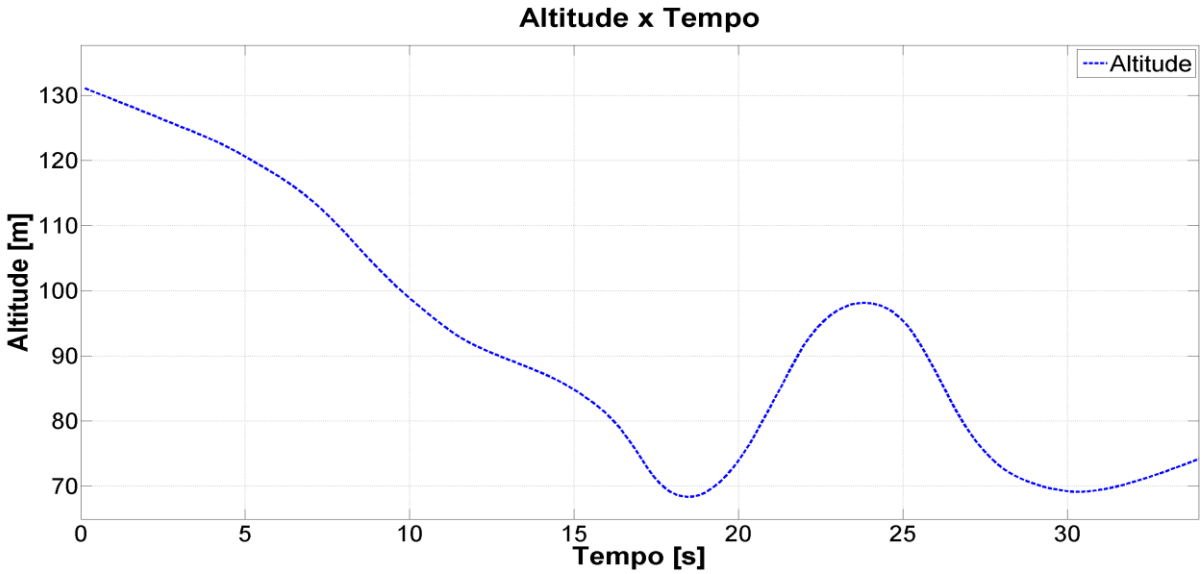


Figura 104 - SIL, MSFS maximizado, Gráfico da Altitude

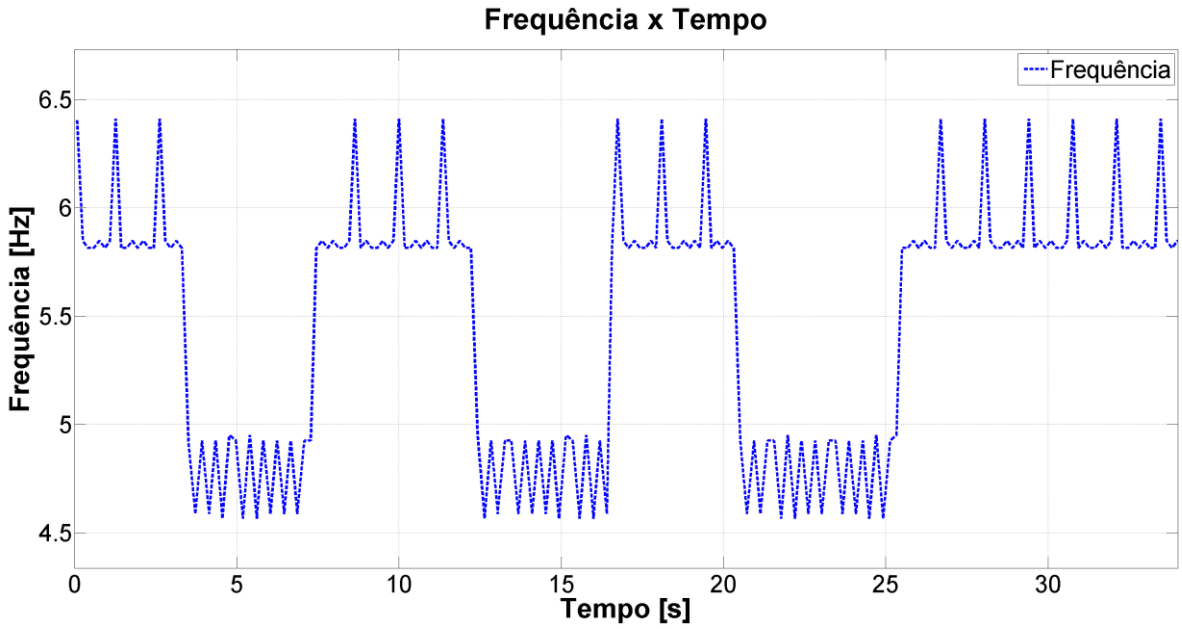


Figura 105 - SIL, MSFS maximizado, Gráfico da Frequência

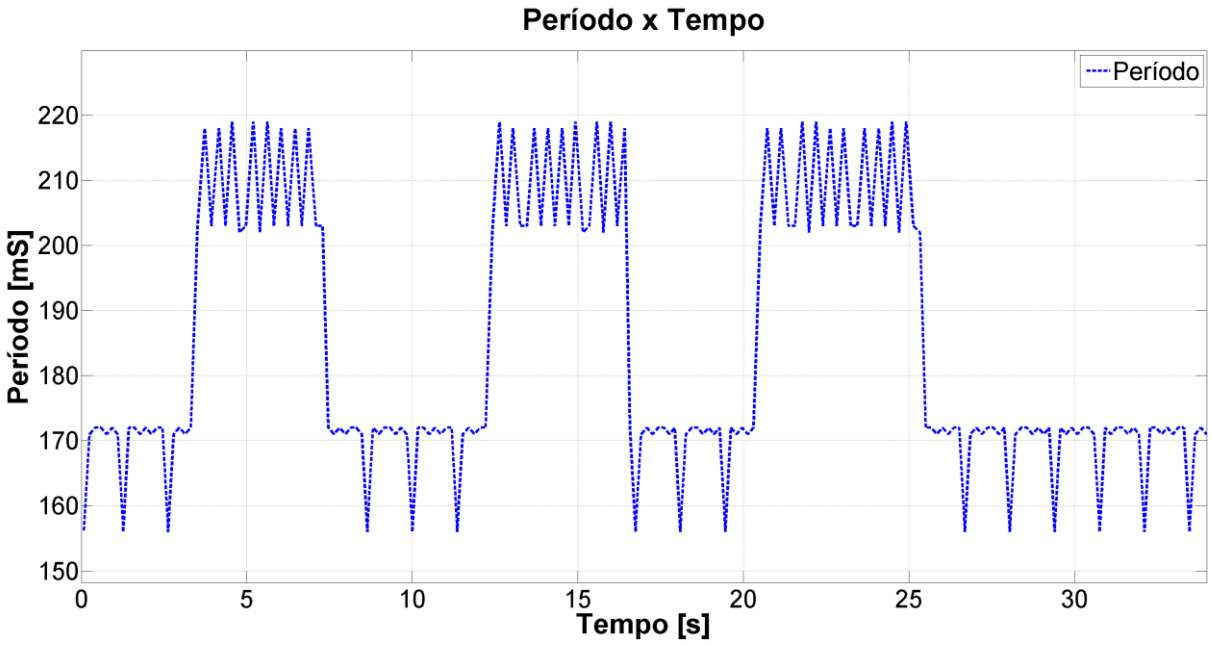


Figura 106 - SIL, MSFS maximizado, Gráfico do Período

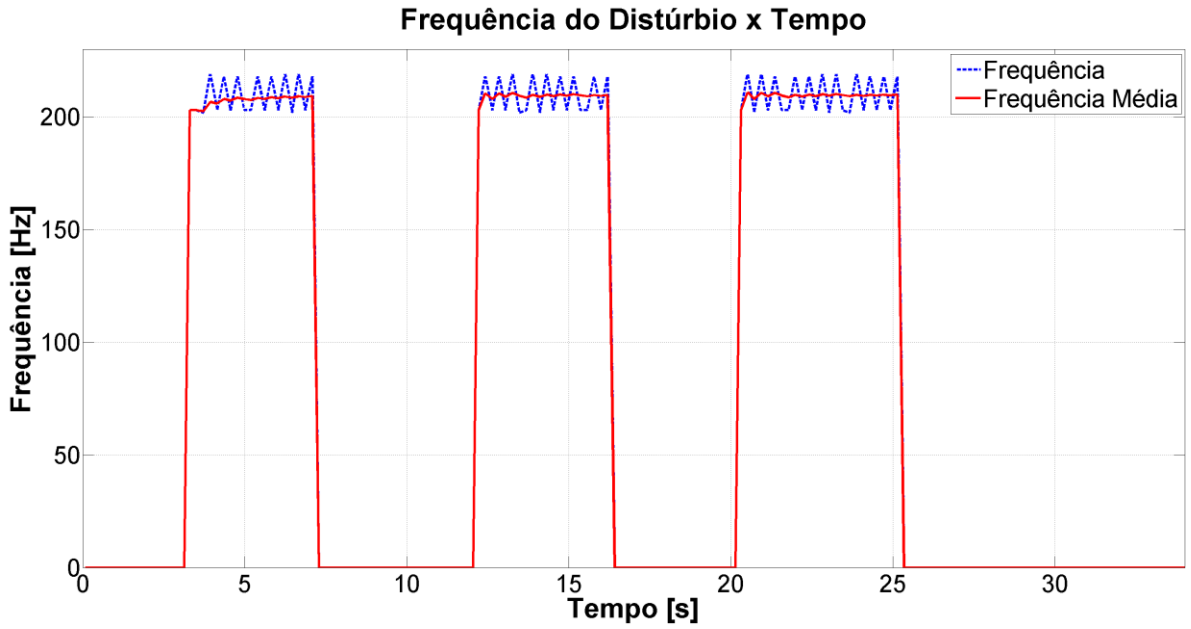


Figura 107 - SIL, MSFS maximizado, Gráfico da Frequência do Distúrbio

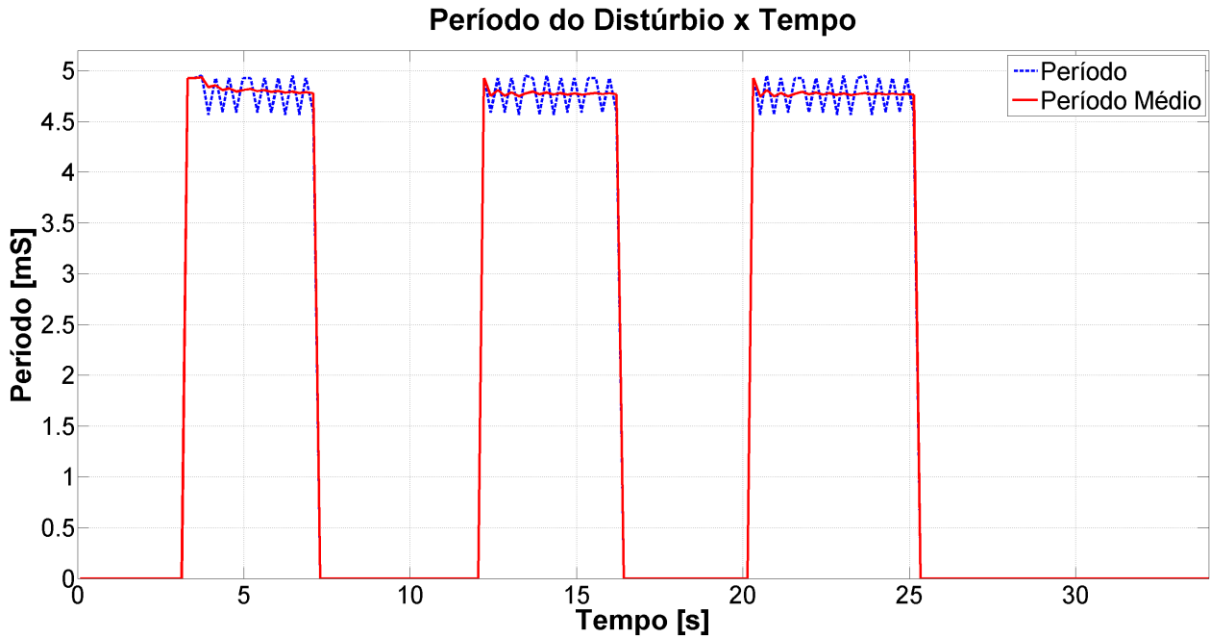


Figura 108 - SIL, MSFS maximizado, Gráfico do Período do Distúrbio

5.6.2. Resultados SIL com MSFS minimizado

A Figura 109 mostra o gráfico de variação dos ângulos de atitude da aeronave durante a aplicação do distúrbio. Note que o valor do ângulo de rolagem, à medida em que o distúrbio era aplicado, não passou de -1,5º. Em contraste ao que aconteceu na Figura 101, observa-se que com a tela minimizada a frequência de atuação do controlador não permite a perda de estabilidade do quadricóptero.

Observe também que, apesar do exposto no parágrafo anterior, os limites de atuação nas superfícies de controle (16383), mais especificamente no *Aileron*, não foram atingidos, como mostra a Figura 111.

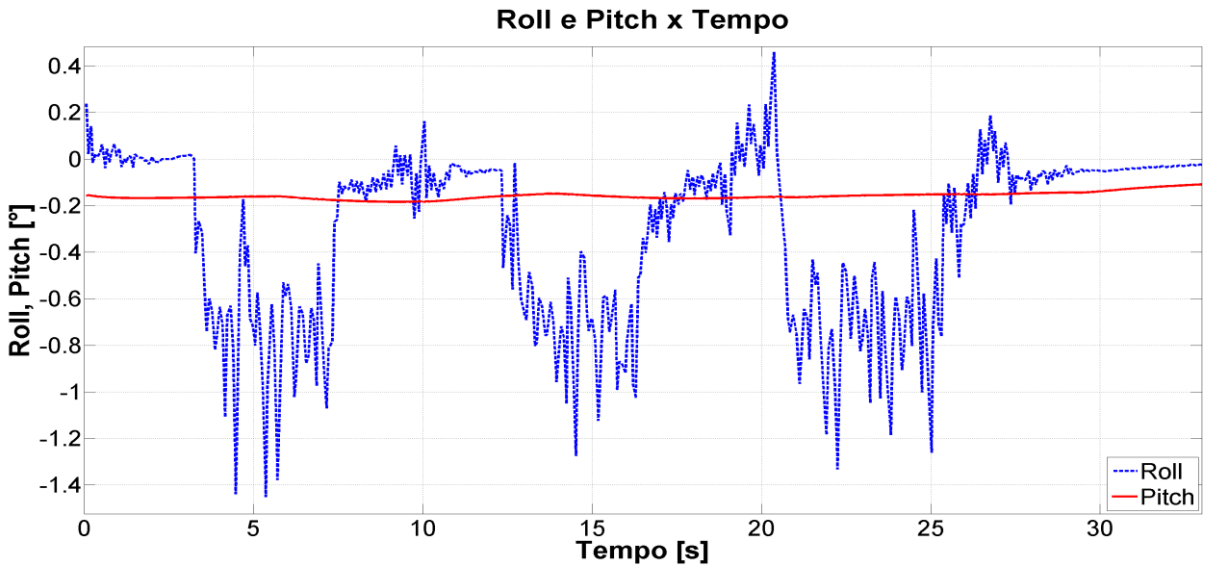


Figura 109 - SIL, MSFS minimizado, Gráfico de Roll e Pitch

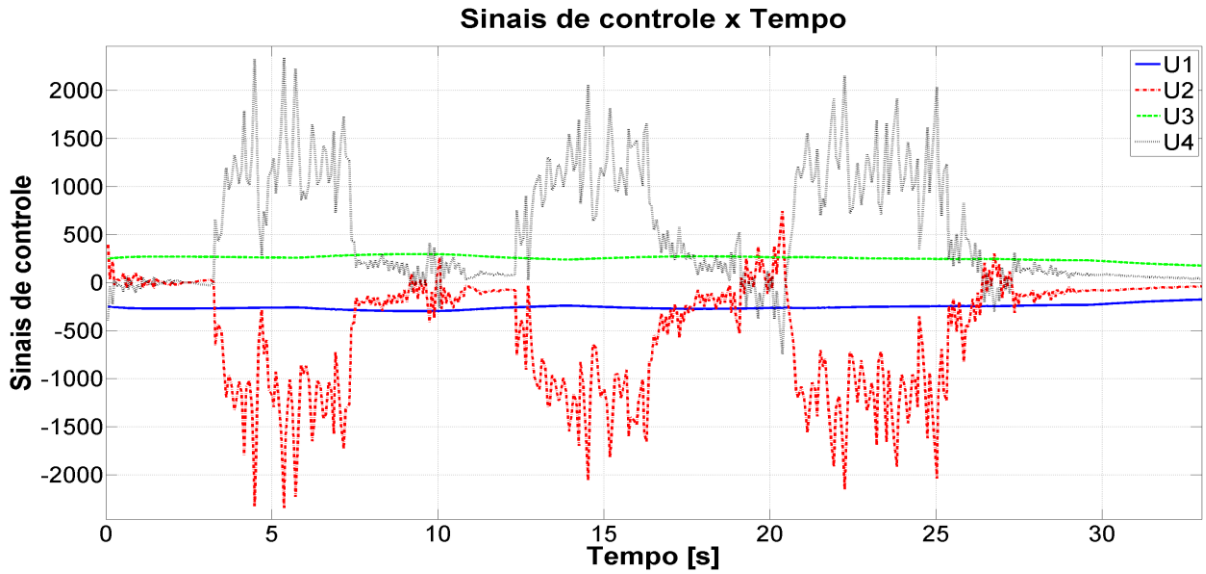


Figura 110 - SIL, MSFS minimizado, Gráfico dos Sinais de Controle

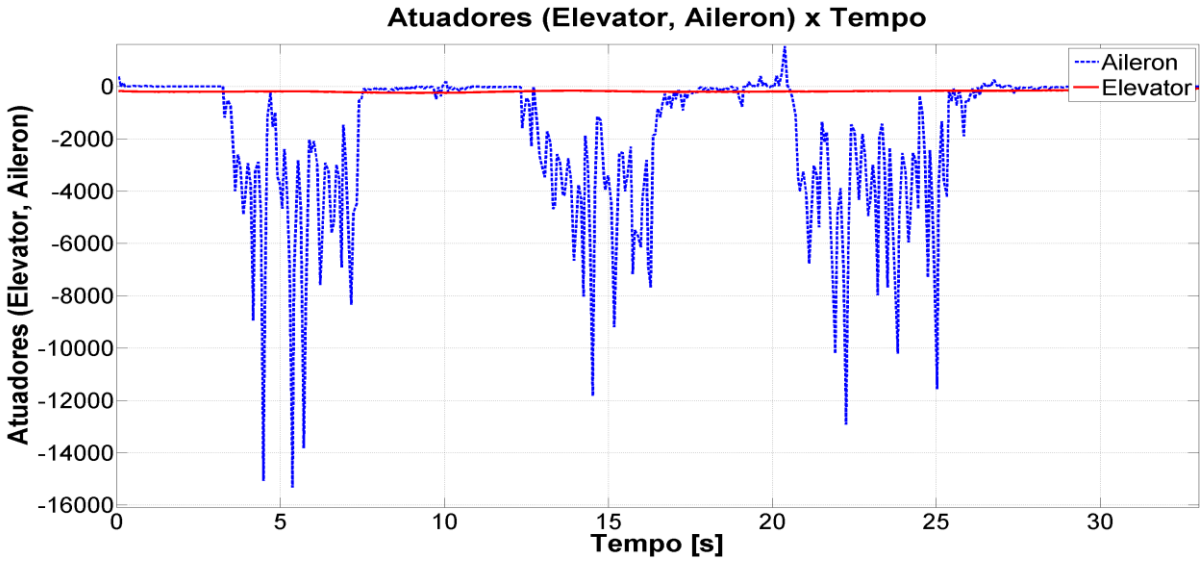


Figura 111 - SIL, MSFS minimizado, Gráfico dos Atuadores

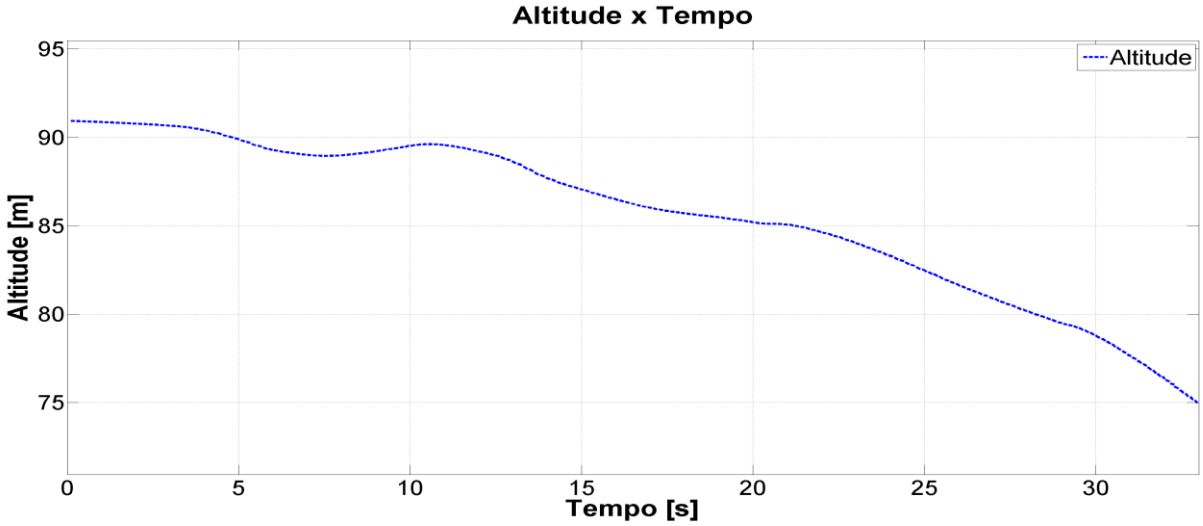


Figura 112 - SIL, MSFS minimizado, Gráfico da Altitude

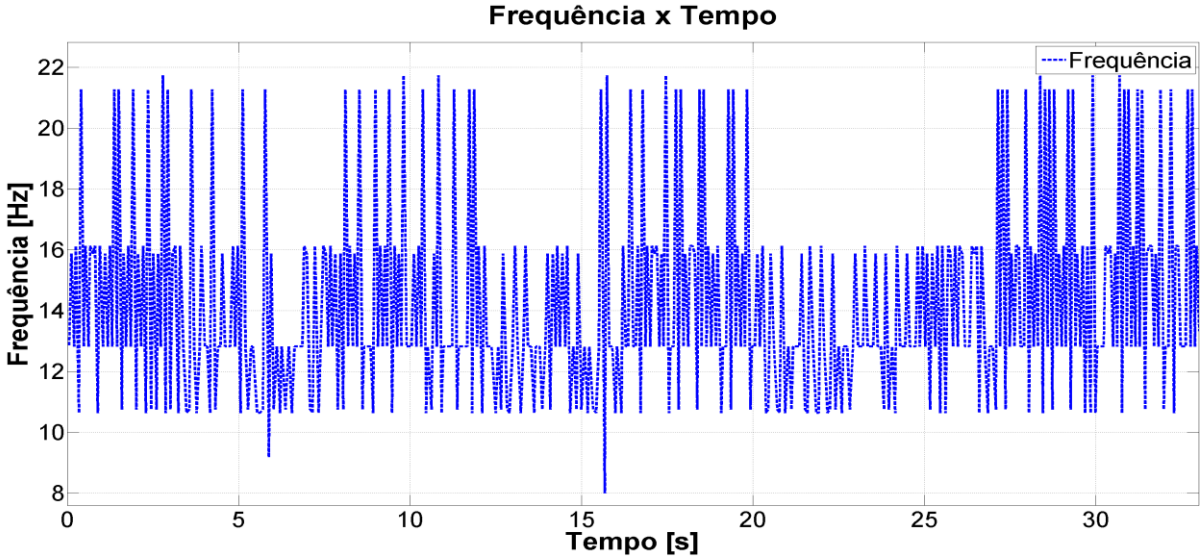


Figura 113 - SIL, MSFS minimizado, Gráfico da Frequência

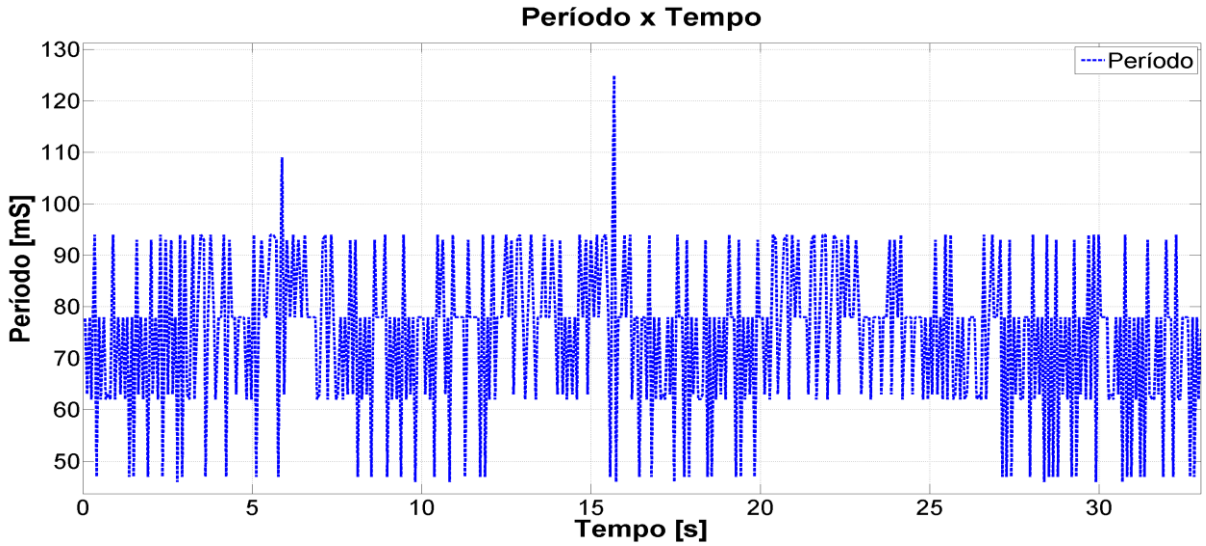


Figura 114 - SIL, MSFS minimizado, Gráfico do Período

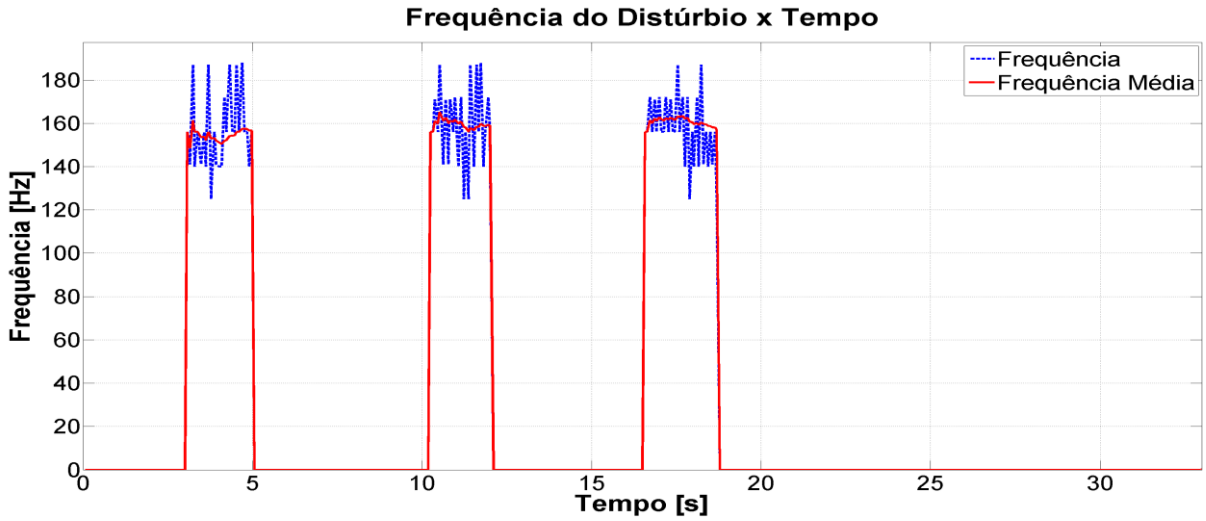


Figura 115 - SIL, MSFS minimizado, Gráfico da Frequência do Distúrbio

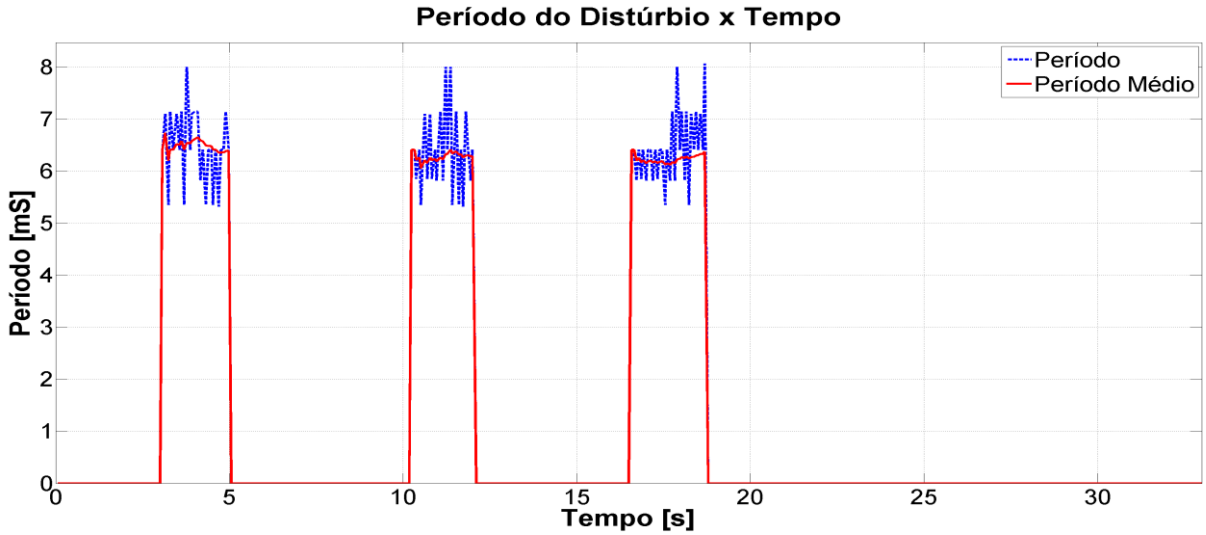


Figura 116 - SIL, MSFS minimizado, Gráfico do Período do Distúrbio

5.6.3. Resultados HIL com MSFS maximizado

A Figura 117 mostra o gráfico de variação dos ângulos de atitude da aeronave durante a aplicação do distúrbio. De forma semelhante ao que aconteceu na Figura 101, observa-se que o controle da aeronave tendia para a instabilidade, resultado da menor frequência de atuação do controlador.

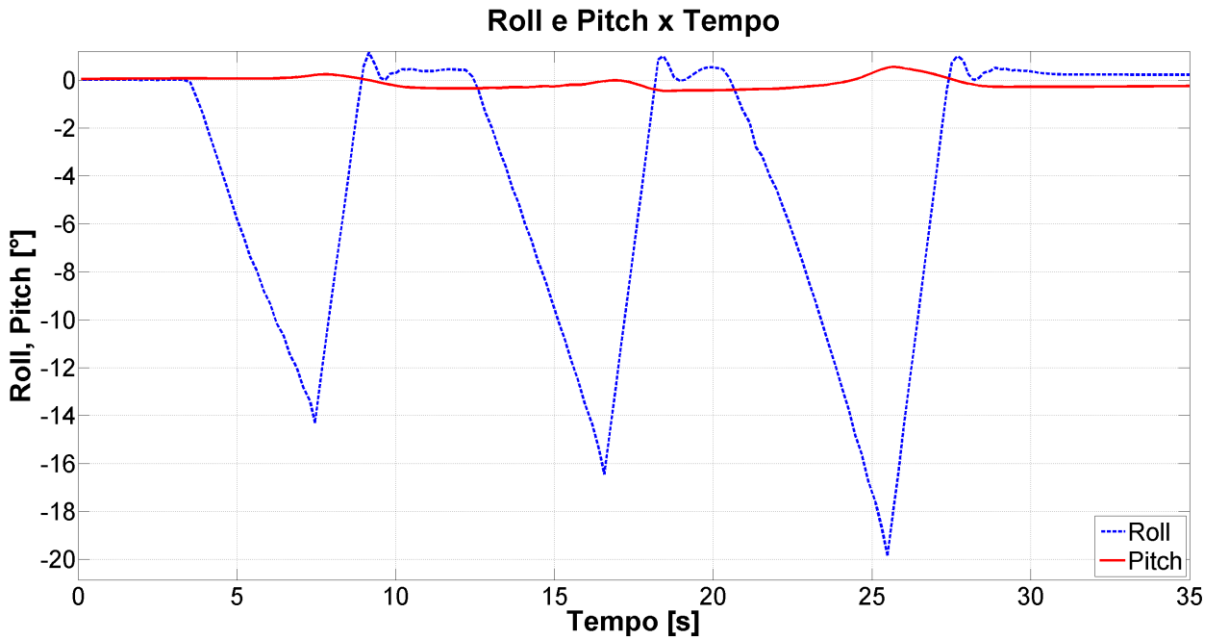


Figura 117 - HIL, MSFS maximizado, Gráfico de Roll e Pitch

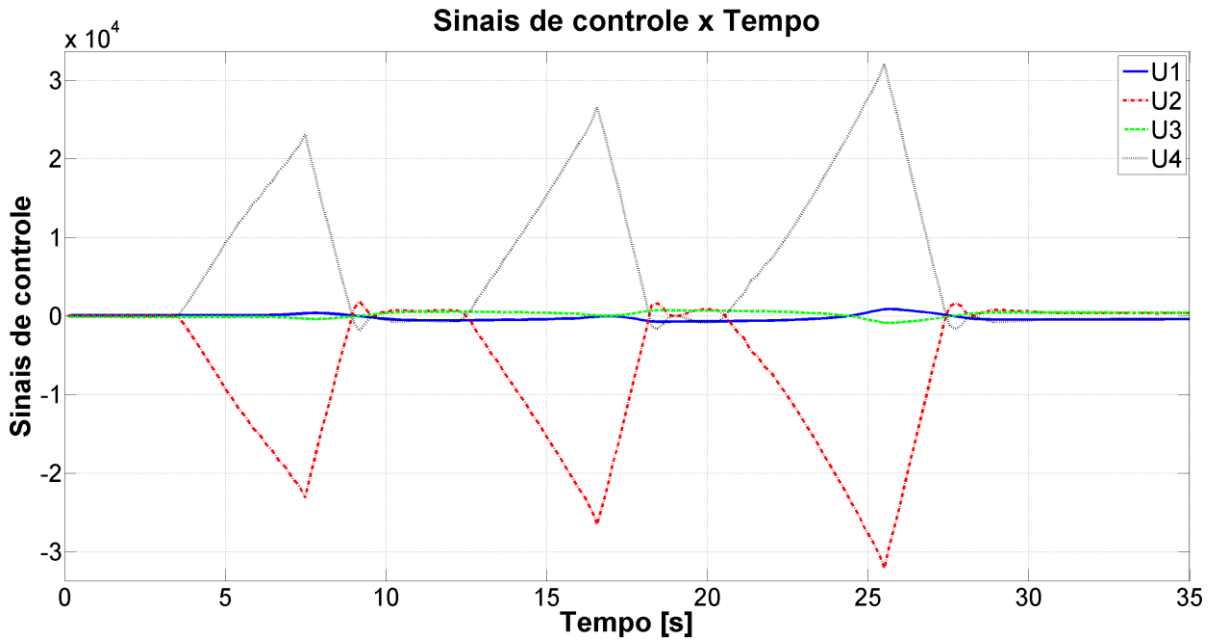


Figura 118 - HIL, MSFS maximizado, Gráfico dos Sinais de Controle

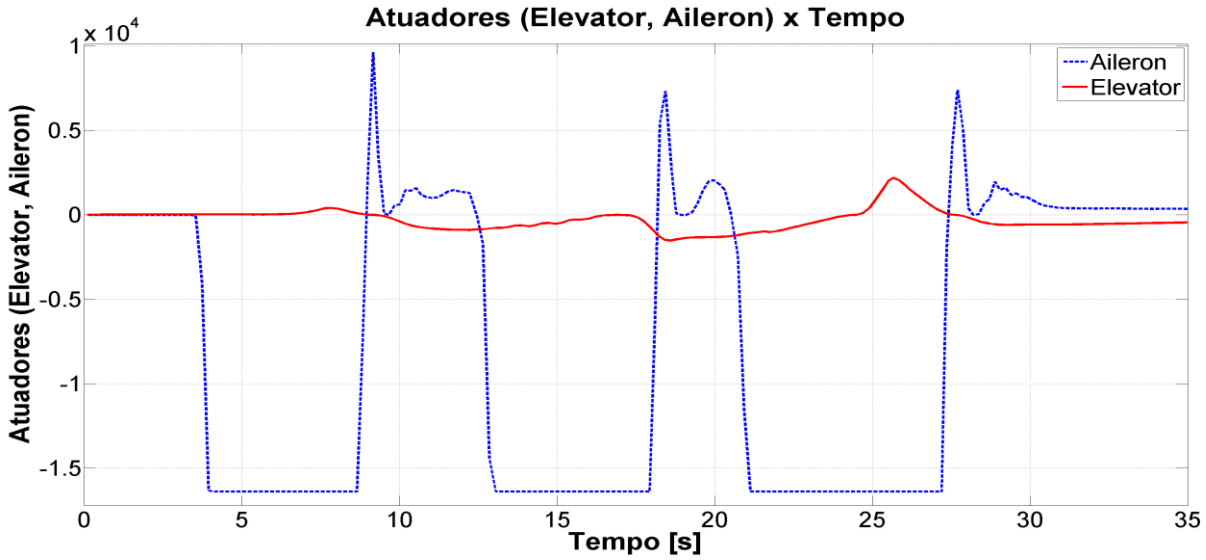


Figura 119 - HIL, MSFS maximizado, Gráfico dos Atuadores

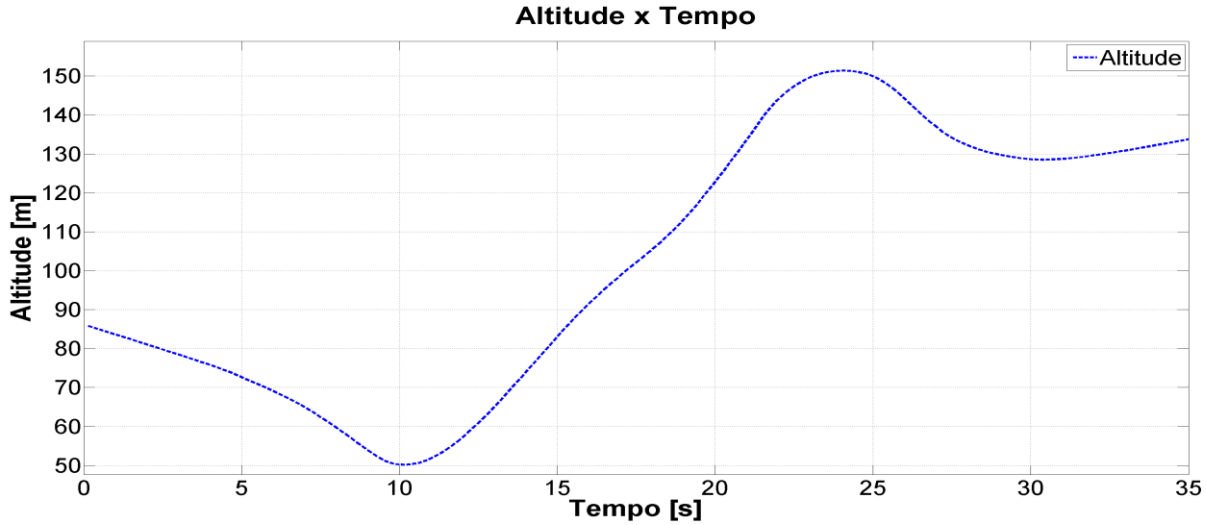


Figura 120 - HIL, MSFS maximizado, Gráfico da Altitude

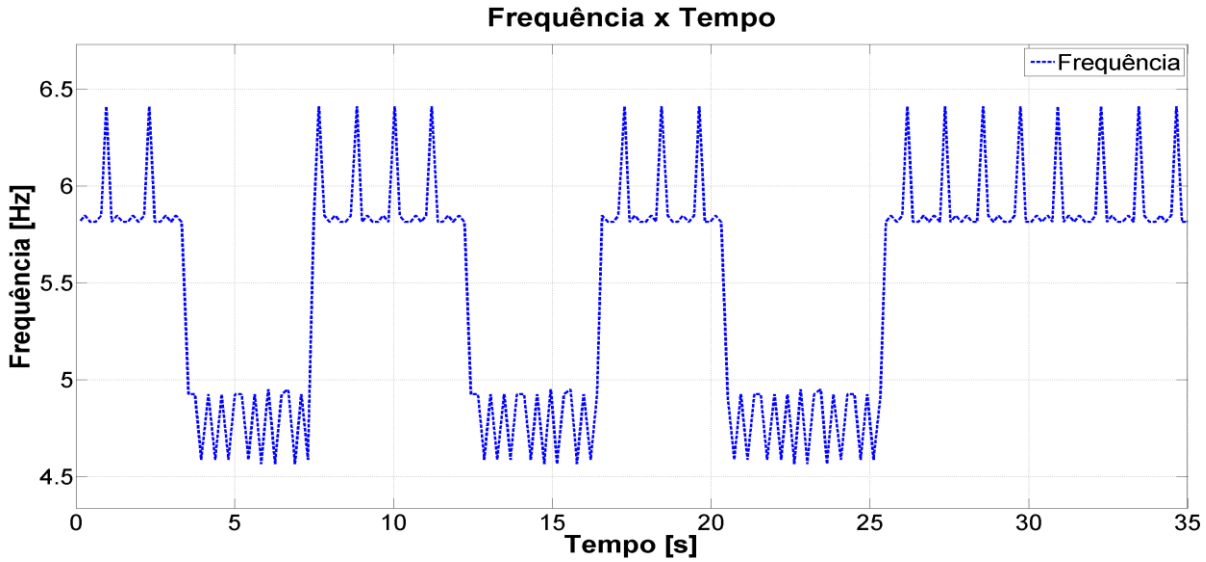


Figura 121 - HIL, MSFS maximizado, Gráfico da Frequência

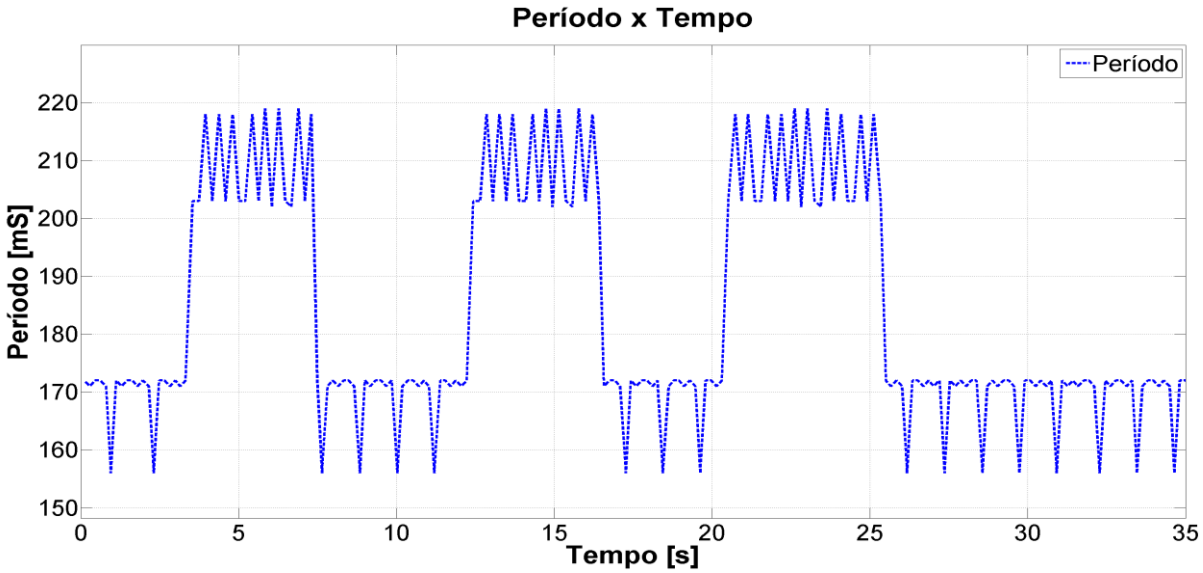


Figura 122 - HIL, MSFS maximizado, Gráfico do Período

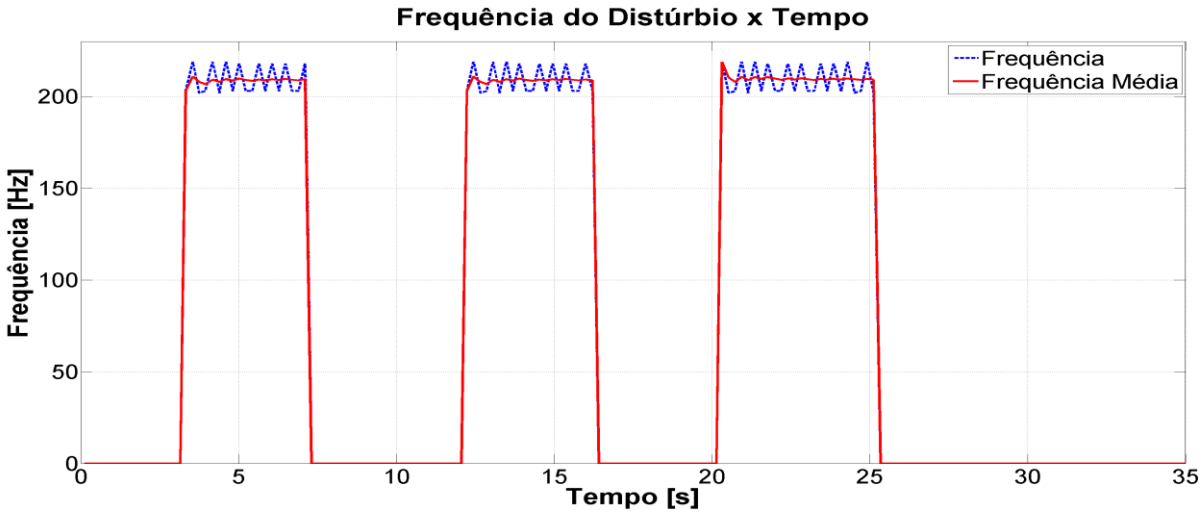


Figura 123 - HIL, MSFS maximizado, Gráfico da Frequência do Distúrbio

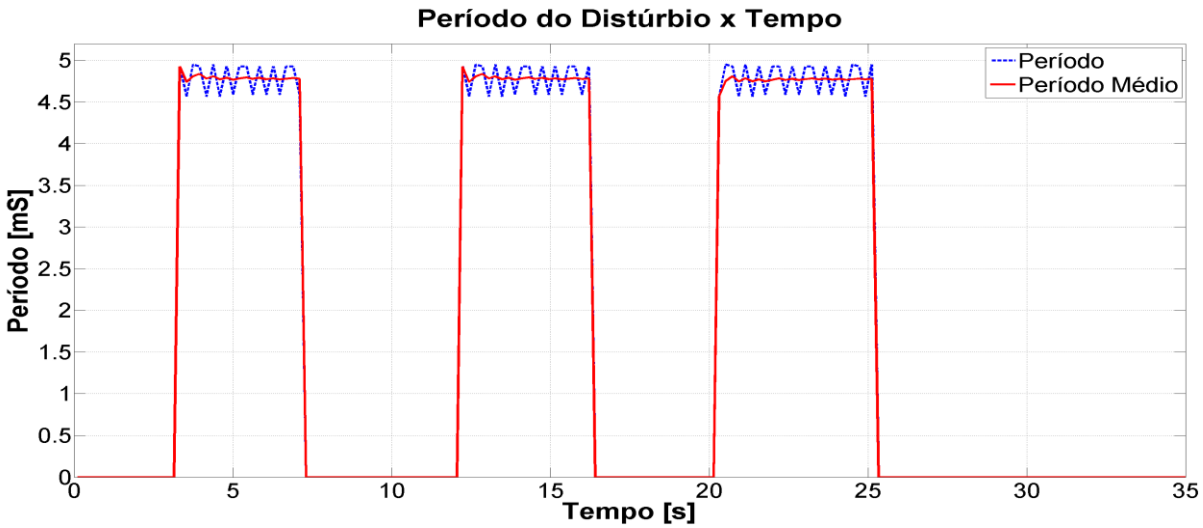


Figura 124 - HIL, MSFS maximizado, Gráfico do Período do Distúrbio

5.6.4. Resultados HIL com MSFS minimizado

A Figura 125 mostra o gráfico de variação dos ângulos de atitude da aeronave durante a aplicação do distúrbio. Assim como ocorreu na Figura 109 (SIL com a tela minimizada), o valor mínimo do ângulo de rolagem não passou de -2,0º, o que permite a conclusão de que a frequência de atuação do controlador não permite a perda de estabilidade do quadricóptero. Diferentemente do ocorrido na Figura 109, no entanto, houve a saturação do valores gravados nos atuadores (16383), conforme mostra a Figura 127. Conclui-se, portanto, que a frequência de atuação do controle está diretamente relacionada com a intensidade dos valores da atuação.

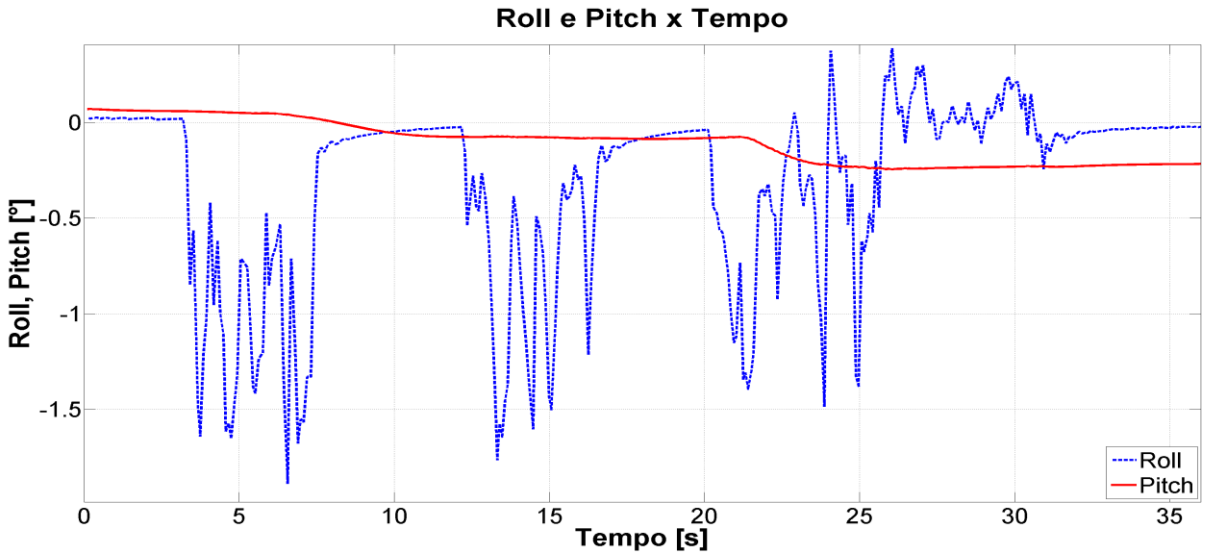


Figura 125 - HIL, MSFS minimizado, Gráfico de Roll e Pitch

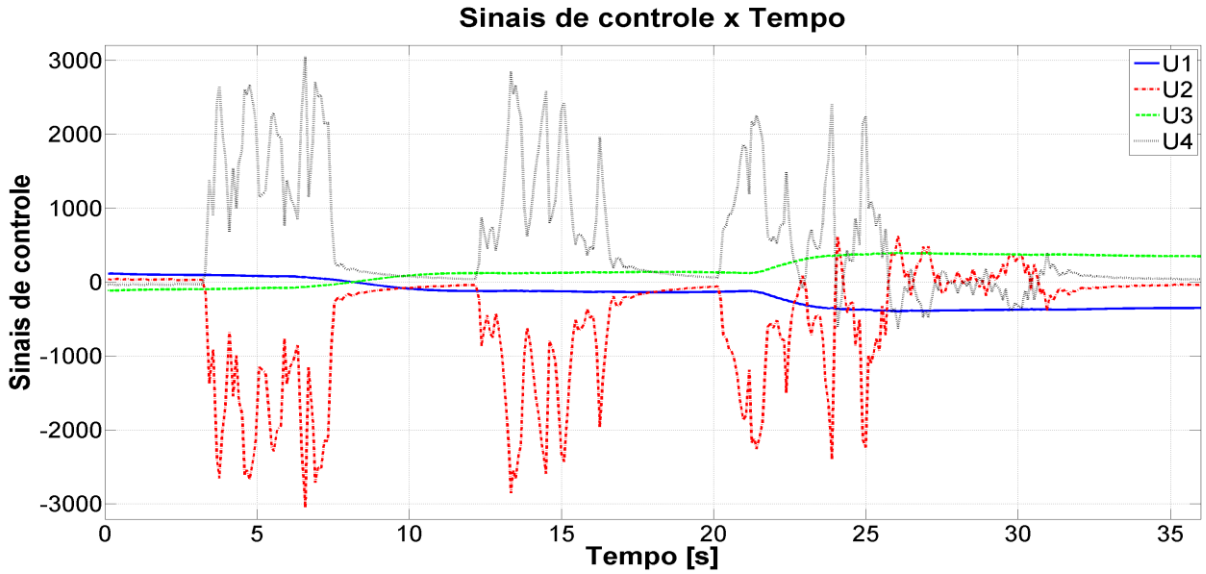


Figura 126 - HIL, MSFS minimizado, Gráfico dos Sinais de Controle

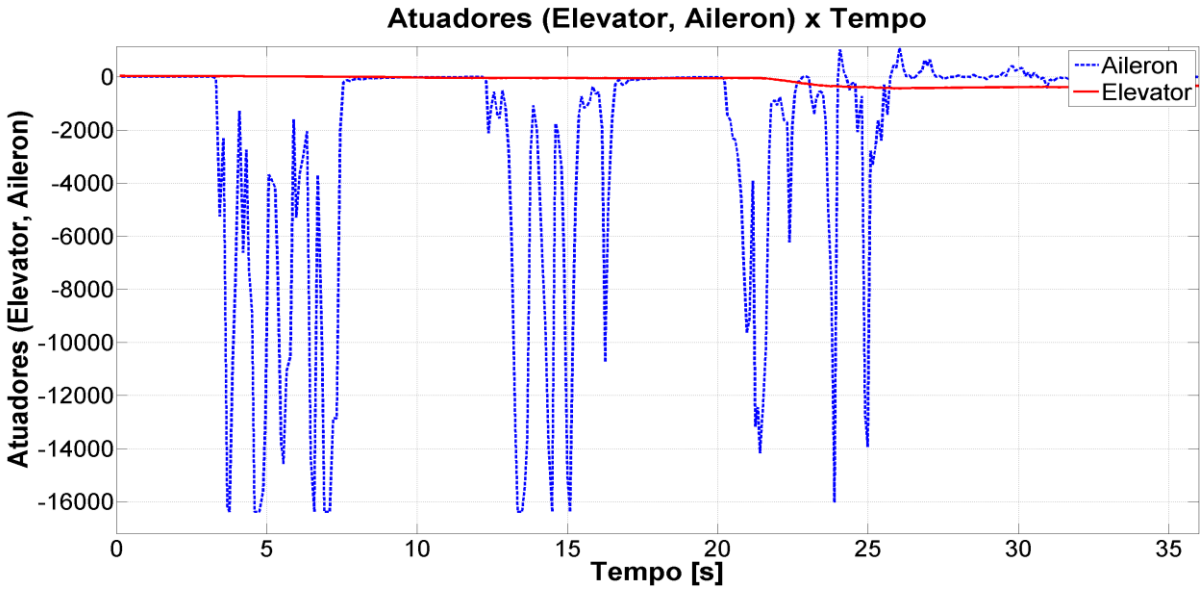


Figura 127 - HIL, MSFS minimizado, Gráfico dos Atuadores

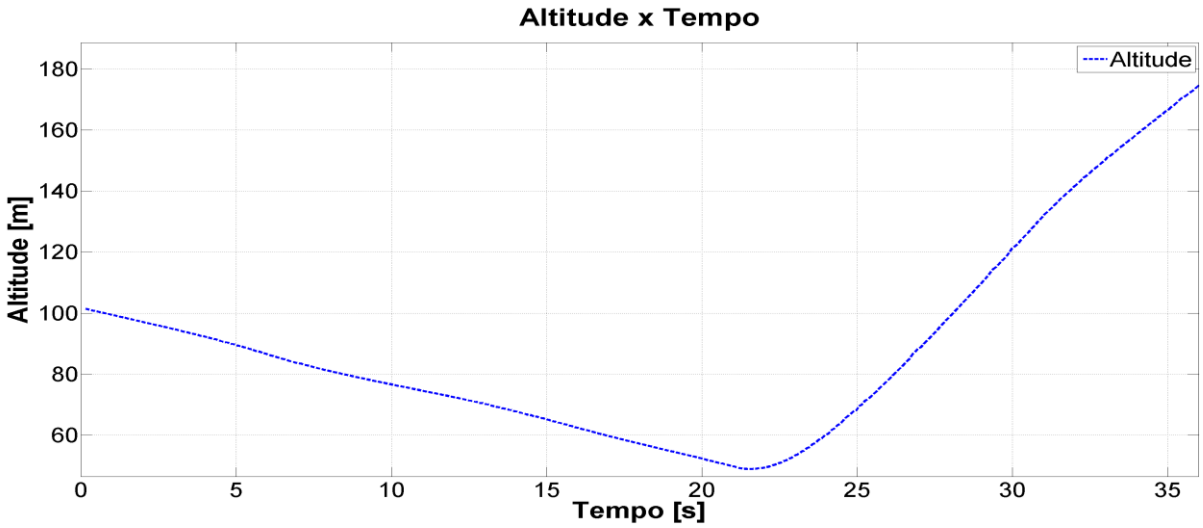


Figura 128 - HIL, MSFS minimizado, Gráfico da Altitude

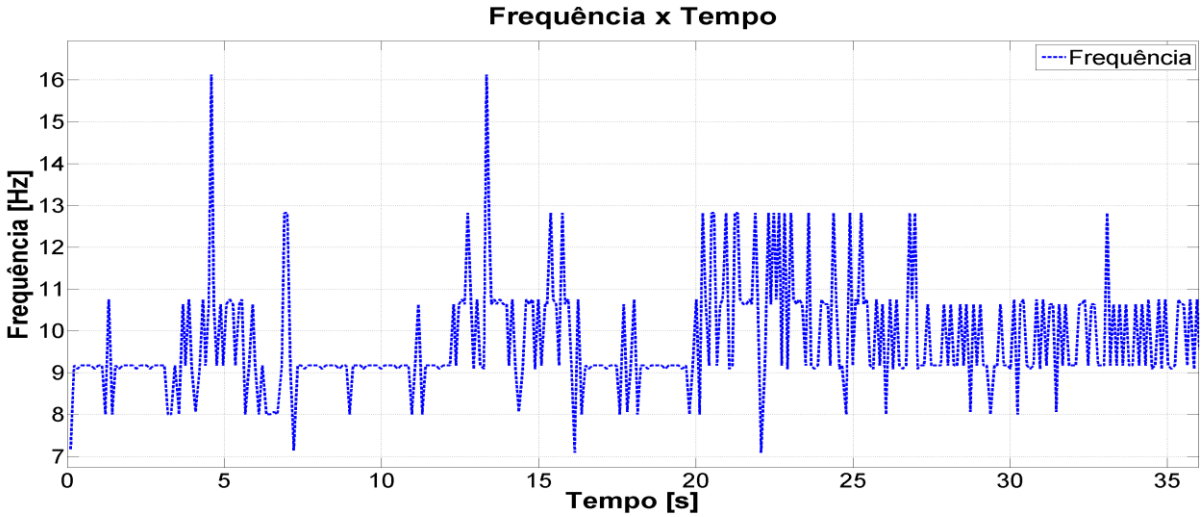


Figura 129 - HIL, MSFS minimizado, Gráfico da Frequência

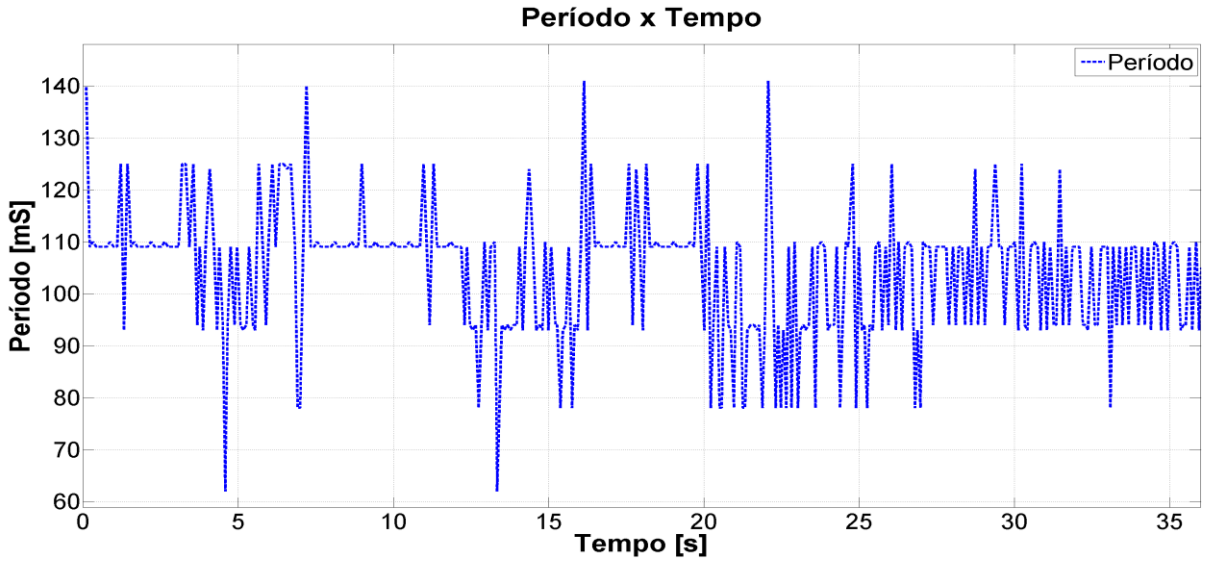


Figura 130 - HIL, MSFS minimizado, Gráfico do Período

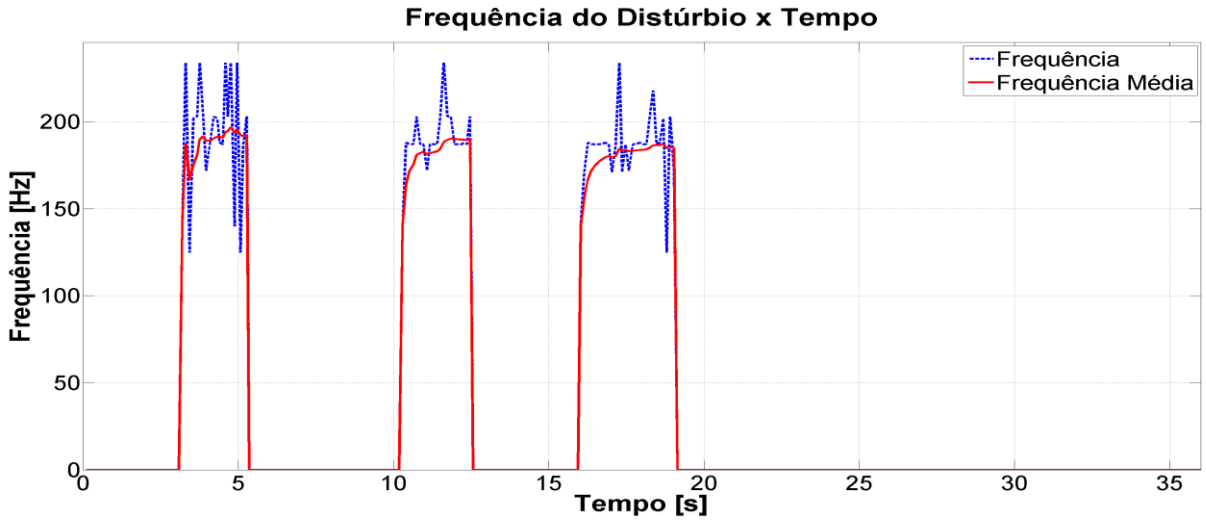


Figura 131 - HIL, MSFS minimizado, Gráfico da frequência do Distúrbio

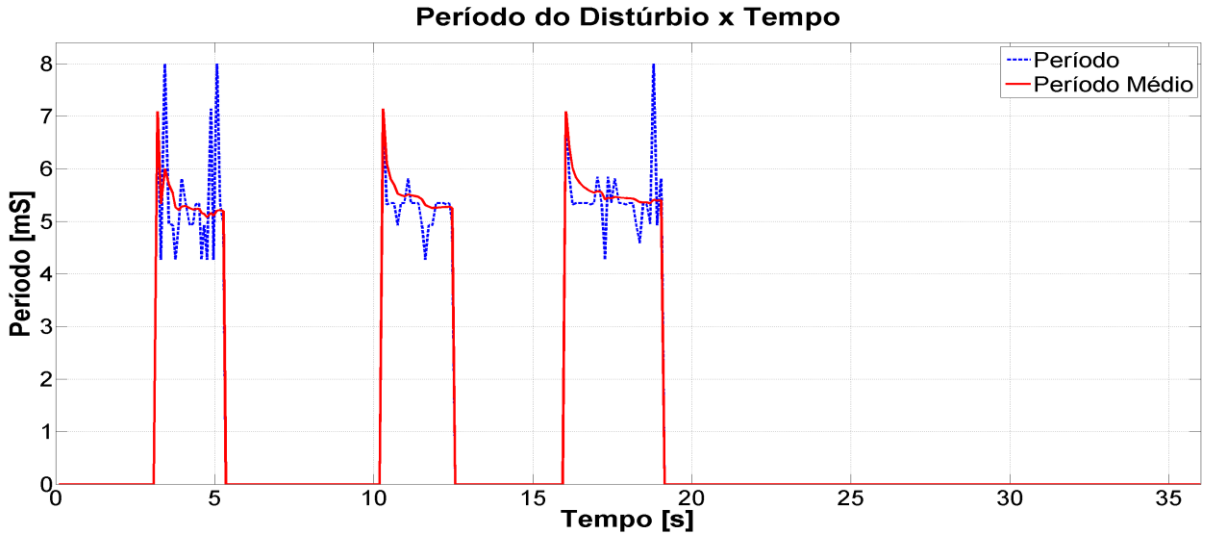


Figura 132 - HIL, MSFS minimizado, Gráfico do Período do Distúrbio

6. Conclusão e Trabalhos Futuros

O objetivo deste trabalho era desenvolver uma plataforma para testes de controladores diversos para o controle de estabilidade de uma aeronave. A plataforma desenvolvida foi constituída por um *software* de simulação de voo (MSFS), um *hardware* externo e um *software* de interface entre o *hardware* e o simulador. De início, acreditou-se que o "gargalo" do sistema estaria na comunicação USB entre o computador e a placa. No entanto, os testes realizados mostraram que o impacto desse tempo era secundário quando comparado com o impacto do tempo de processamento gráfico do MFSF.

De uma forma geral, ficou claro pelo exposto, principalmente no capítulo anterior, que o processamento da imagem pelo *software* prejudicou consideravelmente a eficácia do controle realizado externamente. Nos testes da seção 5.6, ficou claro a superioridade do controle nos casos em que a tela estava minimizada. Nos dois casos em que a tela está maximizada, o avião tendeu a cair quando da aplicação do distúrbio. E nos dois casos em que a tela está minimizada, a frequência de atuação do controle foi suficiente para mantê-lo no ar.

A conclusão a que se chega, portanto, é de que essa plataforma pode ser utilizada para testes preliminares de controladores. No entanto, assim como sugerido por Rozenfeld et al. (2006), o protótipo físico real deve ser adiado ao máximo, porém nunca deve ser descartado no desenvolvimento de um produto.

Mas a questão que fica agora é a seguinte: o que mudaria caso estivéssemos usando um quadricóptero real? Se assim o fosse, o MSFS seria substituído pelo helicóptero e o *hardware* externo seria embarcado no mesmo. A primeira implementação a ser realizada seria a inserção de sensores e atuadores reais. O problema do processamento da imagem deixaria de existir assim como e o tempo de comunicação entre o *hardware* e a planta. Esses tempos seriam substituídos pelos tempos de leitura dos sensores, de resposta dos atuadores, e dos cálculos matemáticos necessários para integração e tratamento dos sinais de alguns dos sensores.

Os principais sensores seriam os constituintes de uma IMU (acelerômetro, giroscópio e o magnetômetro) e outros sensores auxiliares para localização e navegação,

tais como sensor de pressão barométrica, GPS, Sensor laser e o sensor ultrassom. Cada sensor desses possui características de interfaceamento específicas.

Segundo o fabricante, a frequência de atuação do controle do quadricóptero PELICAN é de 1kHz, ou seja, a atualização dos atuadores deve ocorrer a cada 1ms. Este valor está muito abaixo do que conseguimos utilizando o MSFS, porém é quase o dobro do tempo necessário para a realização do controle (Figura 28), 597µs. Nesta linha de raciocínio, conclui-se que a leitura das informações e os respectivos tratamentos deveriam ocorrer em 403µs apenas. Os sensores da IMU (acelerômetro, giroscópio e magnetômetro) encontrados no mercado geralmente possuem comunicação digital (SPI ou I2C) ou analógica (0 a 3.3V). Entre esses dois tipos, a saída digital é preferível à analógica. Em geral, os sensores com saída digital são mais precisos e mais robustos, pois as trilhas de sinais analógicos e o conversor A/D integrados são minúsculos, o que favorece a imunidade a ruídos externos.

A título de ilustração, utilizaremos o componente LSM9DS1 do fabricante ST, ideal para esse tipo de aplicação. Este sensor do tipo MEMS é uma unidade semicondutora que consegue medir aceleração, giro e campo magnético em 3 eixos. A comunicação entre ele e o controlador é digital, SPI ou I2C. Se for escolhida a comunicação por SPI, a frequência do *clock* da SPI pode chegar à 10MHz, resultando num período de 100ns. Se a escolha for pela I_2C, a velocidade é bem menor, variando de 100 a 400Khz, mas ainda sim alta o suficiente para esta aplicação, (LSM9DS1 datasheet). A Tabela 7, mostra que essas velocidades são compatíveis com microcontrolador escolhido para esta aplicação.

Pelos dados de velocidade de comunicação descritos no parágrafo anterior, conclui-se que os 403µs seriam mais do que suficientes. No entanto, a propriedade mais importante desses sensores é a *Output Data Rate* (ORD). A ODR é a taxa de saída das informações geradas pelo sensor. No caso do LSM9DS1, a ODR pode chegar a 952Hz, ou seja, uma atualização de valores a cada 1,05ms. Sendo assim, entre a aquisição de dados dos sensores e a execução do controle, tem-se aproximadamente1,6ms. Mas ainda falta computar eventuais tratamentos desses dados, tais como filtragem, integração e filtro de Kalman, por exemplo. Todos esses procedimentos precisam de tempos bem maiores para sua execução. A leitura do GPS, do sensor laser e do ultrassom também são lentas,

e não devem ser consideradas nesse loop de controle. As informações provindas dessas três fontes devem ser colocadas no vetor de estados desejados da ação de controle.

O microcontrolador escolhido para esse trabalho possui uma parte da memória RAM (2KB) reservada para o DMA (*Direct Memory Access*). O DMA é um recurso que permite um certo grau de independência entre os periféricos do microcontrolador, ou seja, permite por exemplo, que a comunicação com o LSM9DS1 ocorra sem que o programa principal a execute diretamente. Isso faz com que a execução do código da lei de controle ocorra quase que simultaneamente às leituras do sensor.

Uma outra questão a ser considerada refere-se ao erro aleatório desses sensores. Uma técnica comumente empregada, é definir um valor através da média aritmética de um conjunto de leituras. Este procedimento funciona como uma espécie de filtro passa-baixa e tem como objetivo a eliminação de dados incorretos provenientes de erros aleatórios. Mas esse procedimento reduziria ainda mais a frequência de execução do controle.

Considerando os fatores descritos nos parágrafos anteriores, conclui-se que 1ms para tempo de ciclo, como sugerido pela AscTec, seria um tempo muito curto. A sugestão para trabalhos futuros, portanto, seria a avaliação do comportamento real de uma placa semelhante à essa desenvolvida neste trabalho num quadricóptero real.

Referências Bibliográficas

ASCTEC. **Asctec**. Disponivel em: <http://www.asctec.de/>. Acesso em: 10 jul. 2014.

BOUABDALLAH, S.; MURRIERI, P.; SIEGWART, R. Design and control of an indoor micro quadrotor. **International Conference on Robotics & Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference**, New Orleans, LA, abr. 2004. 4393-4398 Vol. 5.

CANTONI, L. F. A.; NETO, A. A.; CHAIMOVICZ, L.; CAMPOS, M. F. M. Análise comparativa entre Microsoft Flight Simulator e Flightgear Flight Simulator em testes hardware-in-the-loop, Brasília, IX Simpósio Brasileiro de Automação Inteligente (SBAI) 2009.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos**. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A, 2009.

DOWSON, P. **Application interfacing module for Microsoft Flight Simulator**, 2014. Disponivel em: <http://www.schiratti.com/dowson.html>. Acesso em: 10 jul. 2014.

DSPIC33FJ128MC706A Datasheet. Disponivel em: <http://ww1.microchip.com/downloads/en/DeviceDoc/70594d.pdf>. Acesso em: 01 jul. 2014.

FELDSTEIN, C. B.; MUZIO, J. C. Development of a fault tolerant flight control system. **The 23rd Digital Avionics Systems Conference IEEE**, 2, 24-28 out. 2004.

FT232BL Datasheet. Disponivel em: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232BL_BQ.pdf>. Acesso em: 01 jul. 2014.

HISTÓRIA do Flight Simulator. **RG Press:** A nova visão do voo virtual, 2014. Disponivel em: <http://rgpress.no.sapo.pt/historiafs.htm>. Acesso em: 10 jul. 2014.

I2C dsPic33. Disponivel em: <http://ww1.microchip.com/downloads/en/DeviceDoc/70000195f.pdf>. Acesso em: 07 dez. 2014.

LEISHMAN, J. G. **Principles of Helicopter Aerodynamics**. 2a Edição. New York: Cambrige University Press, 2006.

LOUALI, R.; BOLLOULA, A.; DJOUADI, M. S.; BOUAZIZ, S. Real-time characterization of Microsoft Flight Simulator 2004 for integration into Hardware In the Loop architecture. **19th Mediterranean Conference on Control and Automation**, 20-23 jun. 2011.

LSM9DS1 datasheet. Disponivel em: <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00103319.pdf>. Acesso em: 07 dez. 2014.

MAHONY, R.; KUMAR, V.; CORKE, P. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. **Robotics & Automation Magazine, IEEE**, v. 19, p. 20,32, set. 2012.

ROZENFELD, H.; FORCELLINE, F. A.; AMARAL, D. A.; TOLEDO, J. C.; SILVA, S. L.; ALLIPRANDINI, D. H.; SCALICE, R. K. **Gestão de desenvolvimento de produtos:** uma referência para a melhoria do processo. 1. ed. São Paulo: Saraiva, 2006.

SAMPAIO, R. C.; BECKER, M.; SIQUEIRA, A. A. G.; FRESCHI, L. W.; MONTANHER, M. P. FVMS: A novel SiL approach on the evaluation of controllers for autonomous MAV. **AIAA/IEEE Aerospace Conference 2013**, Big Sky, Montana, 2013a.

SAMPAIO, R. C.; BECKER, M.; SIQUEIRA, A. A. G.; FRESCHI, L. W.; MONTANHER, M. P. Novel SiL Evaluation of an Optimal H∞ Controller on the Stability of a MAV in Flight Simulator. **AIAA/IEEE Aerospace Conference 2013**, Big Sky, Montana, 2013b.

SAMPAIO, R. C.; BECKER, M.; SIQUEIRA, A. A. G.; FRESCHI, L. W.; MONTANHER, M. P. Optimal H∞ controller on the stability of MAVs in a novel Software-in-the-Loop control

platform. **IEEE International Conference on Industrial Technology (ICIT 2013)**, Cape Town, 2013c. 146.

SAMPAIO, R. C.; HERNANDES, A. C.; BECKER, M.; CATALANO, F. M.; ZANINI, F.; NOBREGA, J. L. E. M.; MARTINS, C. Novel hybrid electric motor glider-quadrotor MAV for in-flight/V-STOL launching. **Aerospace Conference, 2014 IEEE**, Big Sky, MT, 1-8 mar. 2014. 1-12.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Autonomous Mobile Robots**. 2a ed., ed. The MIT Press, 2011.

WIKIPEDIA. **Wikipedia**, 2014. Disponivel em: <https://pt.wikipedia.org/wiki/M%C3%A9todo_de_Monte_Carlo>. Acesso em: 10 fev. 2014.

ZHOU, K.; DOYLE, J. C. **Essential of Robust Control**. Upper Saddle River, New Jersey: Prentice Hall, 1998.