UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Fernando Silvano Gonçalves

PROJETO DA ARQUITETURA DE SOFTWARE EMBARCADO DE UM VEÍCULO AÉREO NÃO TRIPULADO

Florianópolis (SC)

2014

Fernando Silvano Gonçalves

PROJETO DA ARQUITETURA DE SOFTWARE EMBARCADO DE UM VEÍCULO AÉREO NÃO TRIPULADO

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Leandro Buss Becker, Dr. - PGEAS - UFSC
Coorientador: Prof. Guilherme Vianna Raffo, Dr. - PPGEE - UFMG

Florianópolis (SC)

2014

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Gonçalves, Fernando Silvano
Projeto da arquitetura de software embarcado de um veículo aéreo não tripulado / Fernando Silvano Gonçalves ; orientador, Leandro Buss Becker ; coorientador, Guilherme Vianna Raffo. - Florianópolis, SC, 2014.
101 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Sistemas embarcados. 3. Projeto seguro. 4. Anytime Algorithms. 5. Hardware-in-the-loop. I. Becker, Leandro Buss. II. Raffo, Guilherme Vianna. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Fernando Silvano Gonçalves

PROJETO DA ARQUITETURA DE SOFTWARE EMBARCADO DE UM VEÍCULO AÉREO NÃO TRIPULADO

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis (SC), 13 de março 2014.

Prof. Jomi Fred Hübner, Dr.
PGEAS - UFSC
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

---

Prof. Leandro Buss Becker, Dr. - PGEAS - UFSC
Orientador

Prof. Guilherme Vianna Raffo, Dr. - PPGEE - UFMG
Coorientador

Banca Examinadora:

Prof. Leandro Buss Becker, Dr.
PGEAS - UFSC
Presidente

Prof. Cristian Koliver, Dr.
IFSC

Prof. Jean-Marie Alexandre Farines, Dr.
PGEAS - UFSC

Prof. Max Hering de Queiroz, Dr.
PGEAS - UFSC

Aos meu pais Claudi e Juçara, à minha irmã Daniele, à minha namorada Thiana e aos meus amigos pelo apoio concedido.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida, por me conceder sabedoria e ciência, estar comigo em todos os momentos e me guiar em todas as minhas escolhas.

Agradeço também:

A minha amada família, pois sem eles nada disto seria possível, por todo amor, carinho e compreensão concedidos e acima de tudo por sempre me apoiar e incentivar em todas as minhas escolhas, contribuindo tanto para o meu crescimento pessoal, quanto profissional.

A minha namorada Thiana, que apesar de não poder estar sempre por perto, não mediu esforços para me apoiar durante esta jornada. Por todo seu amor, carinho, compreensão, por me oferecer sempre seu ombro amigo e pelas palavras de incentivo que me deram força durante esta caminhada.

Aos meus amigos do projeto ProVant, em especial ao Rodrigo, João Paulo, Patrick, e Martin, pela amizade construída neste período, por todo apoio concedido na realização deste trabalho pois sem a sua ajuda e empenho isto não seria possível. Assim como, pelos momentos de descontração proporcionados ao longo desta jornada.

Aos amigos do Laboratório de Controle e Automação (LCA), em especial a Vinicius e Renato, pelo companheirismo e prontidão, por sempre estarem dispostos a auxiliar e apoiar nas mais diferentes atividades, bem como pela amizade construída.

Aos amigos e colegas de mestrado, em especial a William, Cleber e Rafael, pelo apoio concedido dentro e fora da sala de aula, pelo companheirismo e amizade, e também pelos muitos momentos de descontração compartilhados nesta etapa.

Ao PPGEAS, CAPES, CNPQ e ao CTC pelo apoio, suporte e infraestrutura.

E finalmente agradeço aos grandes incentivadores desta pesquisa, meu orientador Leandro e meu coorientador Guilherme, que não mediram esforços para a conclusão deste trabalho, contribuindo com todo seu conhecimento e amizade.

Enfim, agradeço a todos que de alguma forma contribuíram para a realização desta pesquisa.

Once you have tasted flight, you will forever walk the earth with your eyes turned skyward, for there you have been, and there you will always long to return.

Leonardo da Vinci

RESUMO

O projeto de Veículos Aéreos Não Tripulados (VANTs) é um processo complexo, pois diferentes fatores devem ser considerados visando garantir tanto a confiabilidade do sistema, quanto a integridade física da aeronave. Neste contexto a técnica do *Hardware-in-The-Loop* (HIL) tem sido aplicada de modo a proporcionar uma plataforma de testes confiável e segura para a concepção tanto da arquitetura de software, quanto dos sistemas de controle. Nesta pesquisa um ambiente de emulação em HIL é desenvolvido de forma a proporcionar a realização de testes da arquitetura de software, sendo este baseado na aplicação da técnica dos *Anytime Algorithms*. Esses algoritmos são aplicados tanto isolados quanto associados aos algoritmos de controle do VANT, desta forma os benefícios da sua utilização podem ser corretamente avaliados. O ambiente HIL pode ser utilizado diretamente no Matalab/Simulink, o que facilita o desenvolvimento de diferentes estratégias de controle por meio do uso desta ferramenta.

Palavras-chave: Sistemas Embarcados; Projeto Seguro; *Anytime Algorithms*; *Hardware-in-the-loop*; VANTs.

ABSTRACT

Designing Unmanned Aerial Vehicles (UAVs) is a complex process, since different engineering domains must be considered to ensure that it will fly properly, without putting in danger the aircraft and life on the ground. Considering the design of the embedded computing system, the Hardware-in-The-Loop (HIL) can be applied in order to provide a test platform for both the software architecture and the control algorithms. In this work a HIL structure is created to test the software architecture, which is based in the application of Anytime Algorithms. Such algorithms are used isolated and associated with the UAV control algorithms, so that their benefits for the project can be properly evaluated. The developed HIL can be used directly by the Matlab/-Simulink tool, which facilitates its use since most control applications are developed in such tool.

Keywords: Embedded systems; Safe design; Anytime algorithms; Hardware-In-the-Loop; UAVs.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

A utilização de aeronaves não tripuladas em aplicações civis e militares tem se tornado cada vez mais frequente, sendo estes dispositivos definidos como Veículos Aéreos Não Tripulados (VANTs). Os VANTs são caracterizados pela sua condução sem a ação de tripulação, sendo operados tanto de forma autônoma quanto controlada remotamente. Dentre os ambientes onde estão inseridos podemos destacar aplicações militares, vigilância de tráfego, aplicações voltadas à agricultura de precisão e pesquisas meteorológicas [Budiyono 2008].

Conforme destacado em [Hoffmann et al. 2007], o desenvolvimento de um VANT apresenta diversos desafios de projeto, dentre os quais estão o desenvolvimento do conjunto de funcionalidades e o atendimento aos requisitos não funcionais, tais como, desempenho, custo, consumo de energia e restrições temporais. Nos ambientes onde as plataformas embarcadas se fazem presentes, o atendimento das restrições temporais exige dos engenheiros e projetistas cuidado redobrado nas fases de planejamento e construção, visto que o descumprimento destas pode levar a falhas da aplicação.

Essas aeronaves são caracterizadas por interagir com o ambiente em que estão inseridas, seja na ocorrência dos voos ou nos procedimentos de pouso e decolagem, estando, assim, sujeita às interferências do meio. Esses dispositivos são caracterizados como *Cyber-Physical Systems* (CPS).

Os CPS são sistemas caracterizados pela criação de uma ponte entre o mundo computacional, o qual é responsável pelo processamento das informações, com o mundo físico, onde os processos reais a serem controlados ocorrem [Lee 2008]. Estes sistemas são aplicados a vários ambientes, e podem ser organizados de forma centralizada ou divididos em subsistemas complementares e diferentes [Kim et al. 2012].

O processo de desenvolvimento de um CPS é uma tarefa complexa. Em sua fase inicial de concepção, as dinâmicas físicas devem ser modeladas, e os demais requisitos de projeto devem ser cuidadosamente descritos, visando à manutenção de segurança e confiabilidade das aplicações.

Embora existam diversos modelos de VANTs, sejam eles comerciais ou acadêmicos, a falta de documentação aliada a hardwares e softwares proprietários dificulta muito o processo de reprodução destes projetos. Além disto, as aeronaves comerciais prontas para uso são muito difíceis de personalizar e manter, inviabilizando assim sua aplicação em alguns ambientes.

Neste contexto foi criado no final de 2012 na UFSC em parceria com a UFMG o projeto ProVant[1], que tem por objetivo a modelagem e concepção

[1] http://provant.das.ufsc.br

de um VANT na configuração birrotor para realização de voos de forma autônoma.

O ProVant pretende ser um projeto aberto, visando documentar e apresentar todos os detalhes de projeto do VANT. Dados tanto do projeto eletromecânico quanto do desenvolvimento de algoritmos são disponibilizados tanto no website do projeto, quanto por meio de publicações em eventos relacionados à área.

Este trabalho é parte integrante do ProVant, e descreve o projeto seguro de sistemas embarcados aplicados ao contexto dos VANTs, assim como a sua transição do ambiente de simulação para a plataforma embarcada.

1.1 MOTIVAÇÃO

A concepção de projetos de software aplicados às plataformas embarcadas, bem como a sua aplicação na plataforma alvo, é um processo complexo, o que exige que seu desenvolvimento seja realizado de forma gradual e segura. Nestes ambientes se faz necessário à utilização de uma metodologia de forma a guiar a equipe de projeto e abordar as principais fases da modelagem e concepção destes sistemas, adicionando eficiência e qualidade às etapas desenvolvidas [Becker et al. 2010] [Jensen et al. 2011].

As restrições temporais presentes nessas aplicações são impostas tanto pelo ambiente, quanto pela plataforma embarcada. Esses fatores adicionam um nível elevado de criticidade às aplicações, o que torna o processo de desenvolvimento de sistemas de tempo real aplicado a essas estruturas uma tarefa pouco trivial [Pinheiro et al. 2010].

Buscando garantir o atendimento das restrições impostas, os sistemas de tempo real apresentam técnicas de escalonamento as quais têm por objetivo organizar o conjunto de tarefas de forma que todas sejam executadas utilizando os recursos disponíveis, garantindo, assim, o atendimento aos prazos de maneira que a estabilidade do sistema seja mantida [Liu e Layland 1973].

Baseado na definição de que aplicações de tempo real devem atender aos requisitos temporais, os desenvolvedores buscam tradicionalmente concluir o trabalho solicitado dentro do tempo de computação disponível. No entanto, o cumprimento destes é um processo complexo, o que motivou alguns autores a propor uma nova abordagem, a qual tem por objetivo realizar o trabalho possível dentro do tempo disponível. Ou seja, a qualidade dos resultados obtidos é sacrificada, visando garantir o cumprimento dos prazos estabelecidos. As técnicas que descrevem estas características, são definidas como Computação Imprecisa (do inglês *Imprecise Computation*) [Liu et al.

1994], e visam flexibilizar a execução de aplicações de tempo real.

No contexto da computação imprecisa, diferentes abordagens foram propostas visando à aplicação dessa metodologia, dentre as quais se têm os *Anytime Algorithms* (AA) [Dean e Boddy 1988]. Nesta técnica a solução é construída de forma gradual durante sua execução. Inicialmente se produz um resultado simples, também chamado de impreciso, que é refinado caso haja recursos e tempo de processamento suficiente. Desta forma, caso sua execução seja interrompida, ainda assim uma solução, mesmo que imprecisa, é retornada [Mangharam e Saba 2011].

Devido à criticidade apresentada na concepção dos sistemas aplicados aos VANTs, faz-se necessário o estabelecimento de uma plataforma segura de emulação que descreva as características do ambiente real. Por meio da técnica descrita como *Hardware-In-the-Loop*[2] (HIL), torna-se possível desenvolver e analisar o comportamento destes sistemas.

1.2 OBJETIVOS

O objetivo principal desta dissertação é projetar a arquitetura de software do sistema embarcado de um VANT birrotor, de forma a garantir o cumprimento dos requisitos de tempo real impostos pelo sistema de controle.

1.2.1 Objetivos específicos

1. Pesquisar técnicas de projeto de software voltadas para modelagem de *Cyber-Physical-Systems*.

   Nesta etapa, busca-se analisar as técnicas de projeto de software aplicadas aos CPS, de forma a avaliar a que melhor se enquadra ao escopo do projeto, obtendo assim uma melhor organização, maior clareza das informações e uma abrangência de todos os níveis de projeto.

2. Definir as funcionalidades e projetar a arquitetura do sistema embarcado.

   Para conceber a arquitetura de software embarcado, faz-se necessário o projeto tanto da arquitetura quanto das funcionalidades envolvidas neste ambiente, visando sua aplicação na plataforma alvo. A partir da

[2] O HIL é uma técnica de simulação que combina a plataforma embarcada real juntamente com um modelo computacional do processo, desta forma é possível desenvolver e ajustar algoritmos de controle, e analisar seu comportamento sem colocar em risco o processo real.

análise dos requisitos do sistema é modelada uma proposta de arquitetura de sistema aliada às necessidades dos dispositivos envolvidos.

3. Desenvolver mecanismos visando permitir o projeto seguro de sistemas de controle embarcados de um VANT.

   O desenvolvimento de mecanismos aplicados ao projeto seguro de sistemas de controle embarcados em VANTs visa apresentar estruturas que permitam a validação dos sistemas sem colocar em risco a integridade física da aeronave, bem como das pessoas envolvidas. Assim como, o estabelecimento de um limite seguro de utilização dos recursos embarcados pode ser mensurado, e níveis aceitáveis de confiabilidade e estabilidade da aeronave podem ser assegurados.

4. Avaliar técnicas alternativas de escalonamento de sistemas, visando um aumento da disponibilidade da CPU.

   A partir do ambiente seguro estabelecido, abordagens alternativas de projeto serão avaliadas. Por meio da aplicação da técnica dos AA, o desempenho do sistema, estabilidade, flexibilidade, segurança e confiabilidade são analisados, de forma a validar à factibilidade desta proposta no contexto do VANT.

1.3 ORGANIZAÇÃO

O restante deste trabalho é organizado da seguinte forma: o Capítulo 2 apresenta uma visão geral do projeto Provant, descrevendo os detalhes técnicos da aeronave projetada.

As tecnologias aplicadas no contexto deste projeto, bem como a descrição de alguns trabalhos relacionados ao contexto dos VANTs são apresentados no Capítulo 3.

O Capítulo 4 descreve o processo de projeto seguro de Vants, apresentando desde a sua modelagem funcional até a concepção de plataforma de testes seguras.

Os resultados obtidos com a realização de simulações com os diferentes conjuntos de tarefas, e a análise destes resultados são apresentados no Capítulo 5. Finalizando, o Capítulo 6 apresenta a conclusão e os trabalhos futuros.

1.4 PUBLICAÇÕES

Durante a elaboração dessa dissertação os seguintes trabalhos foram realizados:

- [Goncalves et al. 2013a] GONÇALVES, F. S.; DONADEL, R.; RAFFO, G. V.; BECKER, L. B.; NORMEY-RICO, J. E.. *Small scale UAV with birotor configuration*. In: *International Conference on Unmanned Aircraft Systems* (ICUAS 2013), Atlanta, 2013.

- [Goncalves et al. 2013b] GONÇALVES, F. S.; DONADEL, R.; RAFFO, G. V.; BECKER, L. B.. *Assessing the use of Simulink on the Development Process of an Unmanned Aerial Vehicle*. In: 3º *Workshop on Cyber-Physical Systems* (CyPhy 2013), Filadélfia, 2013.

- [Goncalves et al. 2013c] GONÇALVES, F. S.; RAFFO, G. V.; BECKER, L. B.. *Analyzing the Use of Anytime Algorithms on an Unmanned Aerial Vehicle*. In: 3º Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC 2013), Niterói, 2013.

2 PROJETO PROVANT

O escopo do ProVant é composto por três dissertações de mestrado: o desenvolvimento de diferentes estratégias de controle aplicadas ao contexto do VANT, desenvolvidas pelo mestrando Rodrigo Donadel; a integração da aeronave com a estação base, assim como com uma rede de sensores sem fio, desenvolvido pelo mestrando João Paulo Bodanese; e o projeto seguro de sistemas embarcados descrito neste trabalho.

O projeto seguro de sistemas embarcados compreende o desenvolvimento de alguns componentes, dentre as quais temos a modelagem da arquitetura de software embarcado e o estabelecimento de uma plataforma de simulação que descreva as características do ambiente real.

A partir desses componentes, o atendimento de algumas características do sistema se torna possível, tais como, o atendimento às restrições temporais, e a definição de um limiar de execução de segurança do sistema.

O processo de desenvolvimento do VANT descreve alguns passos, realizados de forma a permitir a concepção da aeronave, onde temos: a definição dos objetivos de projeto; a especificação do hardware embarcado; a representação das dinâmicas da aeronave; a modelagem funcional do sistema; o projeto da plataforma de simulação; e a avaliação do comportamento dos algoritmos de controle propostos. Esses procedimentos são guiados por uma metodologia de desenvolvimento de projeto sendo descritos nas próximas seções.

Na concepção da aeronave, optou-se por desenvolver um VANT birrotor VTOL (*Vertical Take-Off and Landing*), de configuração tiltrotor[1], o qual tem por características a realização de decolagens e pousos na vertical. A Figura 1 apresenta o protótipo inicial adotado.

A proposta principal do projeto é permitir que o VANT realize voos autônomos de acordo com trajetórias pré-definidas. Estas trajetórias (conjunto de *waypoints*) são fornecidas pela estação base, sendo definidas na fase de planejamento das missões. A comunicação entre o VANT e a estação base é realizada por meio de um link *wireless* (Figura 2).

Por meio da interface de software disponível na estação base, o usuário pode realizar as seguintes funções: definir os objetivos das missões e os pontos que o VANT deve seguir na sua execução; iniciar e abortar as missões; monitorar as informações tanto do voo quanto da execução da missão; e realizar testes de verificação dos sensores e atuadores.

Para realização do voo autônomo é necessário que uma missão seja previamente planejada e carregada no VANT, e que o usuário da estação base

[1] O tiltrotor é uma aeronave caracterizada pela movimentação por meio da atuação de seus rotores. Seu deslocamento ocorre devido à inclinação (do inglês *tilt*) de seus rotores.

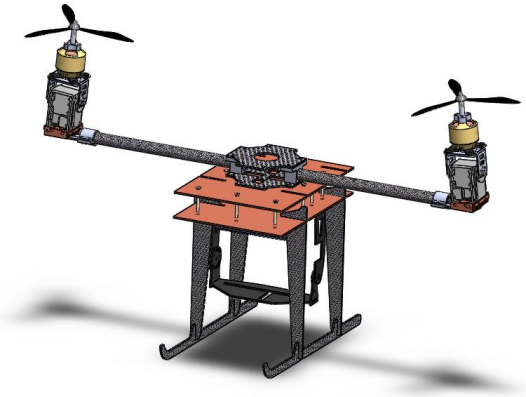

Figura 1 – Protótipo ProVant com configuração tiltrotor.

autorize o início da sua execução. No entanto, além do carregamento da missão, alguns subsistemas devem ser executados no VANT, tais como: controle de velocidade dos rotores, controle do ângulo de inclinação do conjunto propulsor, controle de estabilidade, controle de seguimento de trajetória, monitoramento do nível de carga de bateria, procedimentos de pouso e decolagem e a detecção e evitamento de obstáculos.

Durante a execução, uma missão pode ser abortada em decorrência de dois possíveis eventos, por opção do usuário da estação base ou pela ativação do próprio sistema embarcado. A ocorrência automática por parte do sistema embarcado pode ser ocasionada devido a mudanças nas condições meteorológicas, ou a falhas no sistema. Desta forma, duas ações distintas podem ocorrer na ativação deste modo.

A primeira dessas é denominada retornar a base (do inglês *Return-To-Home* (RTH)), onde o VANT retorna a uma coordenada geográfica específica, normalmente próxima à estação base. Na ativação deste modo o subsistema de controle de missão da aeronave faz a geração de uma nova trajetória de forma a retornar a sua posição inicial. Já a segunda ação consiste na realização de um pouso de emergência no local seguro mais próximo da sua localização atual.

Testes de verificação do equipamento do VANT podem ser realizados como ação preventiva, similares aos testes realizados pelos pilotos antes de alçarem voo. Sua realização dispara uma série de procedimentos responsáveis por verificar a integridade do sistema e da comunicação com a estação base.

Além dos modos de operação descritos, o VANT também pode ser operado por meio de um rádio controle. Ele pode ser ativado tanto para a

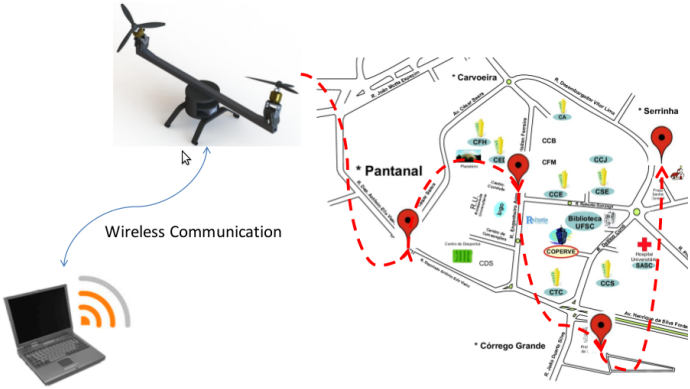

Figura 2 – Descrição da proposta do sistema.

realização de voos, quanto invocado pelo usuário da estação base a qualquer momento no decorrer de uma missão.

Visando suportar os voos e garantir os dados necessários para estimação do seu comportamento e posicionamento, o VANT necessita de um conjunto de sensores e atuadores responsáveis por dar suporte ao sistema embarcado.

2.1 SENSORES E ATUADORES

Para a realização dos voos, o VANT executa algoritmos de navegação. Navegação é o campo de estudo responsável pelo planejamento e execução dos movimentos do veículo. Nesse processo é necessário a implementação de um sistema de localização que permita a obtenção da posição, velocidade e orientação da aeronave em relação às coordenadas do sistema geográfico [Olivares-Mendez et al. 2010].

Visando aprimorar a estimação da localização, sensores redundantes são utilizados de forma a aumentar a precisão do algoritmo de estimação e reduzir o ruído e falhas nas medições. Esse conjunto de sensores que compõem a estrutura do VANT foi especificado com base nas necessidades de estimação da postura do sistema e no custo destes equipamentos. Sua estrutura apresenta diferentes protocolos de comunicação. Dessa forma, é necessário integra-la à plataforma embarcada de forma a permitir seu correto funcionamento.

Os seguintes sensores são utilizados no projeto ProVant: um sonar; um sistema de posicionamento global (do inglês *Global Positioning System*

- GPS); e uma unidade de medição inercial (do inglês *Inertial Measurement Unit* - IMU), sendo esta composta por um acelerômetro de três eixos, um giroscópio de três eixos, um magnetômetro de três eixos e um barômetro.

O GPS recebe as informações via satélite e calcula a posição e a velocidade por meio de triangulação, visto que ele conhece a distância entre os satélites. Para realização da estimação é necessário que o sinal de pelo menos três satélites esteja disponível.

Na configuração do VANT, o GPS LEA-6h, produzido pela Ublox, é utilizado. Ele fornece uma sensibilidade aceitável de -162 dBm para rastreamento e navegação, e uma precisão de 2.5 m para posição horizontal, 0.1 m/s para velocidade e 0.5 *graus* para posição (Figura 3). Além disso, o receptor necessita de apenas 26 *segundos* para iniciar. Os dados do GPS são atualizados a uma taxa de 2 Hz sendo transmitidos via comunicação serial [uBlox 2013].

Figura 3 – GPS LEA-6h.
Fonte: UBlox (2014).

A estimação da atitude[2], altitude e aceleração do VANT é realizada por meio da IMU STEVAL_MKI124V1, produzida pelo STMicroelectronics, (Figura 4). Esse dispositivo se comunica com a plataforma embarcada via protocolo *Inter-Integrated Circuit* (I2C) e apresenta um conjunto de dados de saída de 16$bits$ do acelerômetro, magnetômetro e giroscópio e um conjunto de dados de saída de 24$bits$ do barômetro.

O acelerômetro permite que suas medições sejam configuradas com diferentes níveis de sensibilidade, sendo estas $\pm 2g$, $\pm 4g$, $\pm 8g$ e $\pm 16g$. A sensibilidade da velocidade angular também pode ser configurada entre os valores 250, 500 e 2000 $graus/s$. A altitude é obtida por meio do barômetro, o qual obtém o valor da pressão absoluta que é comparada com o modelo de variação da pressão relativa à superfície. A mensuração do barômetro varia entre 260 e 1260 *bar* [Steval 2013].

[2] A atitude de um veículo descreve a orientação deste com relação a um sistema de coordenadas de referencia.

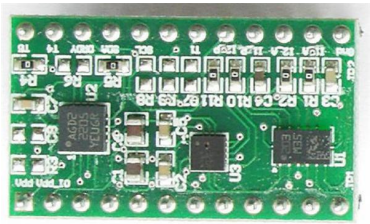

Figura 4 – IMU Steval MKI124V1.
Fonte: STMicroelectronics (2014).

Para melhorar a estimativa de altura em baixas altitudes é utilizado um sonar. No VANT é empregado o modelo MB1200 fabricado pela Maxbotix que fornece medições entre 20 e 765*cm* com 1 *cm* de resolução e uma taxa de amostragem de 10 *Hz* (figura 5) [Maxbotix 2013].

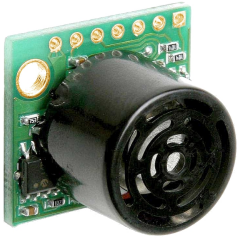

Figura 5 – Sonar MB1200.
Fonte: Maxbotix (2014).

Além dos sensores responsáveis pela estimação da posição, o VANT necessita que forças sejam produzidas em diferentes direções para seu deslocamento. Elas são geradas por meio dos atuadores, os quais são controlados por sinais gerados pelos algoritmos de controle visando a execução da missão atribuída.

O VANT na configuração tiltrotor possui dois servomotores e dois grupos de propulsão, compostos pelo motor *brushless*, pela hélice e pelo controlador do motor.

Os servomotores utilizados são do modelo Dynamixel RX-24F fabricados pela Robotis. Eles são controlados via comunicação serial RS 485. Por meio da atuação destes componentes, os rotores podem ser rotacionados longitudinalmente em 180*graus*, permitindo, assim, que o VANT se mova em diferentes direções [Robotis 2013].

A força responsável por movimentar o VANT é proporcionada pelo

conjunto composto do motor *brushless* e da hélice. O motor embarcado é o modelo AXI 2814/20 produzido pela Modelmotors. Utilizando esse componente é possível levantar uma carga total de $1,3kg$ [Modelmotors 2013].

Estes motores são operados por meio de um controlador de velocidade (do inglês *Electronic Speed Control* - ESC) de modelo BL-Ctrl v2.0 fabricado pela Mikrokopter (Figura 6). Esse dispositivo permite o interfaceamento direto com o motor, onde a partir do valor de referência de velocidade recebido, a velocidade de rotação do motor é estabelecida. Além do controle de velocidade, esse dispositivo envia dados de medição à plataforma embarcada tais como corrente, temperatura, tensão entre outros [Mikrokopter 2013]. A comunicação com a plataforma embarcada é realizada por meio do protocolo I2C.

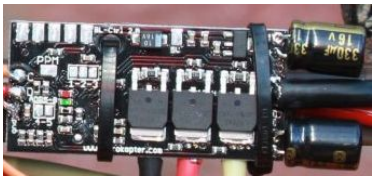

Figura 6 – Controlador *brushless* BL-CTRL V2.0.
Fonte: Mikrokopter (2014).

O receptor de rádio wireless utilizado é o modelo MRF24J40MC (Figura 7), produzido pela Microchip e que implementa o padrão IEEE 802.15.4. Esse padrão contém um conjunto de especificações as quais definem um protocolo adequado para conexão de diferentes dispositivos em uma rede sem fio pessoal (do inglês *Wireless Personal Area Network - WPAN*). O protocolo define as camadas física (PHY) e de controle de acesso ao meio (MAC). Tal padrão é utilizado como base para protocolos de comunicação como ZigBee, MiWi, 6LowPan, WirelessHART, entre outros [IEEE 2011].

O MRF24J40MC é um receptor de $2.4GHz$, e possui uma baixa taxa de dados (250 $kbps$ ou $625kbps$) no modo proprietário. Ele integra as camadas física e de acesso ao meio funcionalmente em uma mesma solução. Além disso, ele conta com um amplificador de potência e um amplificador de baixo ruído, onde por meio da utilização de uma antena externa é possível atingir um alcance de até $1200m$ [Microchip 2013].

Os componentes são gerenciados por uma estrutura de sistema composta por duas camadas. Uma responsável pelo interfaceamento com os sensores, enquanto a segunda realiza um controle de mais alto nível de forma a disponibilizar uma estrutura capaz de estabelecer comunicação com a estação base e executar os algoritmos de controle.

Figura 7 – Rádio MRF24J40MC.
Fonte: Microchip (2014).

2.2 PLATAFORMAS EMBARCADAS

O dispositivo de mais alto nível na estrutura do VANT é a plataforma embarcada Beaglebone (Figura 8), uma plataforma de baixo custo, tamanho similar ao de um cartão de crédito e equipada com um processador superescalar ARM Cortex-A8 de 720MHz aliado a um sistema operacional Linux [BeagleBoard 2013].

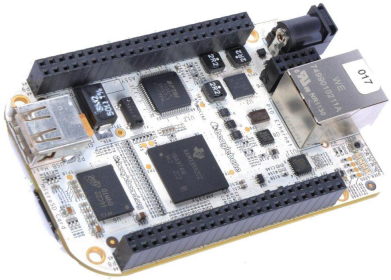

Figura 8 – Plataforma embarcada Beaglebone.
Fonte: Beagleboard (2014).

Esta plataforma é responsável pela execução dos algoritmos de controle, os quais têm por objetivo manter a estabilidade da aeronave no decorrer de todos os procedimentos, e garantir o seguimento de trajetória com base na missão especificada.

Um sistema operacional Linux embarcado compõe esta estrutura, fornecendo um ambiente com maior nível de abstração, permitindo assim o interfaceamento tanto com a outra plataforma embarcada, quanto com os sensores. Esta estrutura apresenta suporte a diferentes protocolos de comunicação,

tais como serial, *Serial Peripheral Interface* (SPI) e I2C. A distribuição Linux utilizada na Beaglebone é o Angstrom.

O Angstrom Linux foi iniciado por um pequeno grupo de pessoas dos projetos OpenEmbedded, OpenZaurus e OpenSimpad, onde estes propuseram a unificação de esforços visando o desenvolvimento de uma distribuição estável e de boa usabilidade voltada a plataformas embarcadas [OpenSimpad 2013]. Este sistema tem suporte a diferentes arquiteturas tais como *handhelds, set top boxes* e dispositivos de armazenamento de rede.

Uma segunda plataforma embarcada é utilizada juntamente com esta estrutura de alto nível, visando a comunicação dedicada com os sensores e atuadores. A adição de um segundo dispositivo de gerenciamento foi realizada com base na análise das características dos sensores e atuadores, bem como de acordo com os tempos de respostas exigidos de cada um destes. Dessa forma, a necessidade do uso de uma plataforma adicional foi verificada, incluindo-se assim a STM32-H407.

A STM32-H407 (Figura 9) é uma plataforma embarcada de baixo custo produzida pela Olimex, a qual possui um processador ARM Cortex M4, aliado ao sistema Linux FreeRTOS [Olimex 2013].

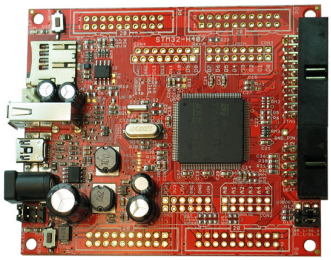

Figura 9 – Plataforma embarcada STM32-H407.
Fonte: Olimex (2014).

O FreeRTOS, um sistema operacional de tempo real (do inglês *Real Time Operational System - RTOS*), desenvolvido pela Real Time Engineers Ltd, visando a aplicação em microcontroladores. Seu objetivo principal é prover um sistema com alto nível de qualidade, robustez e controlabilidade aplicado a essas arquiteturas. Atualmente 33 plataformas são suportadas.

Visto que esse dispositivo não realiza a execução dos algoritmos de controle, nem recebe interferência de nenhum outro subsistema, pode-se então garantir uma maior segurança e confiabilidade na comunicação com os sensores e atuadores, os quais são vitais para o correto funcionamento do VANT.

A estrutura definida pelas plataformas embarcadas, sistemas operacionais embarcados e o conjunto de sensores e atuadores é apresentada na Figura

10. Para a correta comunicação deste conjunto, o desenvolvimento de alguns algoritmos se faz necessário. O projeto desses sistemas não é uma tarefa trivial. Desta forma, é necessário que uma estrutura seja planejada visando à implantação nas plataformas embarcadas. Esses sistemas são descritos na Seção 4.2.

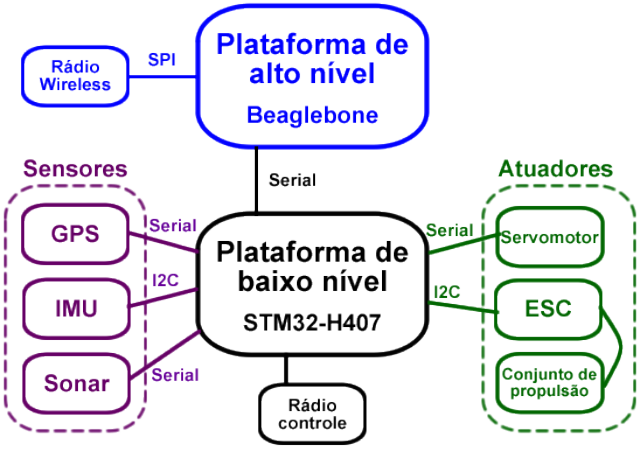

Figura 10 – Estrutura do sistema embarcado.

Assim como o planejamento das rotinas de software, a representação matemática das dinâmicas do sistema é descrita. Desta forma, o modelo matemático aliado à arquitetura de software permite implementar um modelo computacional do VANT e simular seu comportamento. Sua representação matemática é descrita a seguir.

2.3 REPRESENTAÇÃO DAS DINÂMICAS DA AERONAVE

As características dinâmicas da aeronave, assim como as forças e os torques aplicados são representados por meio de expressões matemáticas no contexto do VANT. Com base nelas, estratégias de controle podem ser projetadas visando a estabilização de seu comportamento durante o voo e o seguimento de trajetória.

A estrutura da aeronave é descrita na Figura 11, ela é composta por dois rotores os quais podem ser rotacionados longitudinalmente em 180 graus pela atuação do servomotor. Além disto, os grupos de propulsão são inclinados lateralmente por um ângulo fixo β. Assim, mesmo não havendo atuado-

res para modificar esta inclinação lateral dos rotores, a controlabilidade neste eixo é alcançada [Raffo et al. 2011].

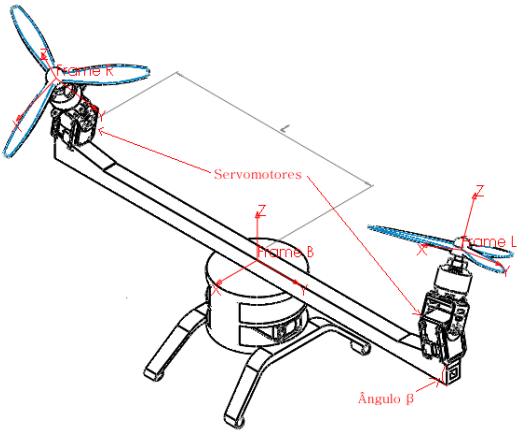

Figura 11 – Protótipo do VANT com configuração tiltrotor.

A inclusão desta inclinação fixa β apresenta como limitação a redução do empuxo vertical máximo que o veículo pode gerar, diminuindo assim a eficiência da aeronave. No entanto, considerando a utilização de um ângulo β pequeno (em torno de cinco graus) o ganho apresentado em controlabilidade deste sistema mecânico subatuado compensa essa perda.

Outra característica de desenvolvimento importante é o deslocamento intencional do centro de massa no eixo Z^B, visando melhorar o torque de arfagem (τ_θ). Este deslocamento é realizado posicionando a bateria, a qual corresponde ao componente mais pesado da aeronave, no nível mais baixo da estrutura central.

Conforme apresentando na Figura 11, três estruturas são consideradas: a estrutura principal B, o rotor da direita R e o rotor da esquerda L. As inclinações longitudinais causadas pelos servomotores são descritas pelos ângulos α_R para o rotor da direita e α_L para o rotor da esquerda. As estruturas R e L rotacionam juntamente com seus rotores, de forma que os eixos Z^R e Z^L são sempre coincidentes com os respectivos empuxos.

A representação matemática referente às forças do VANT, as equações que descrevem os torques em torno dos eixos X, Y e Z e o comportamento da aeronave são detalhados em [Donadel et al. 2014].

Com base no detalhamento do comportamento do VANT, a partir da representação das forças e torques, busca-se projetar estratégias de controle para este sistema, de forma que a aeronave possa realizar voos de maneira

autônoma. Neste contexto, a estrutura de controle adotada compreende o desenvolvimento de três módulos interdependentes: o controle rotacional, controle translacional e transformação de sinais, conforme apresentado na Figura 12.

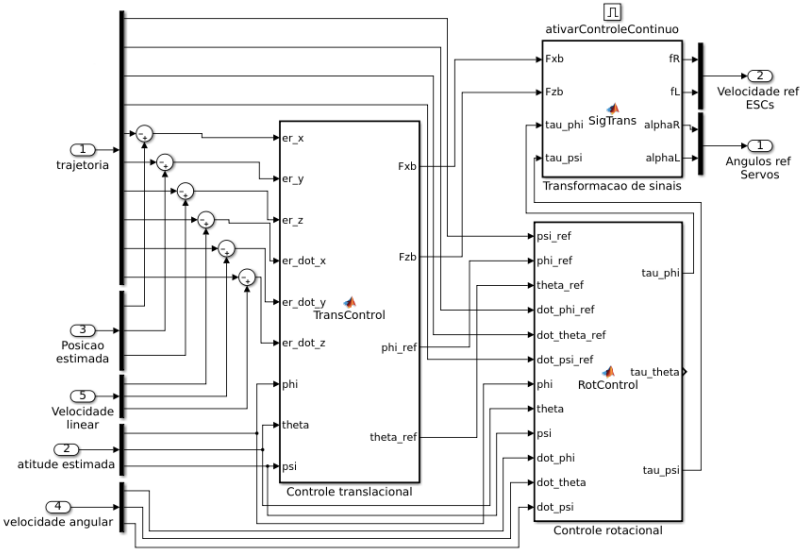

Figura 12 – Módulo de controle contínuo.

O controle translacional é responsável pelo seguimento de trajetória da aeronave. Este é baseado na posição da aeronave (X, Y e Z), assim como em sua velocidade linear ($\dot{X}$, $\dot{Y}$ e $\dot{Z}$), aliado à orientação da mesma (representado por meio dos ângulos ϕ, θ e ψ). Com base nesses valores, são geradas as forças f_{xb} e f_{zb}, assim como, os ângulos de referência ϕ_{ref} e θ_{ref}. As forças e os ângulos de referência calculados são inseridos como entradas do módulo de controle rotacional.

O controle rotacional compõe o segundo bloco da malha de controle, sendo responsável pela estabilização do VANT durante o voo. O cálculo deste ocorre com base nos seguintes parâmetros: na orientação da aeronave (ϕ, θ, ψ); nas velocidades angulares ($\dot{\phi}$, $\dot{\theta}$ e $\dot{\psi}$); nos ângulos de referência (ϕ_{ref} e θ_{ref}) gerados pelo controle translacional; no ψ_{ref} fornecido pelo gerador de trajetória; e nas velocidades angulares de referência ($\dot{\phi}_{ref}$, $\dot{\theta}_{ref}$ e $\dot{\psi}_{ref}$).

Devido ao fato de que as saídas dos blocos de controle rotacional e translacional são descritas em grandezas distintas das entradas dos atuadores do sistema, esses valores devem ser devidamente convertidos para que

possam ser enviados aos atuadores da aeronave. Dessa forma, esses dados são inseridos como entrada no terceiro bloco, responsável pela conversão em valores de referência aplicados aos atuadores do sistema.

Este subsistema recebe como entrada as referências de força f_{xb} e f_{zb} e os valores de referência angular τ_{phi}, τ_{theta} e τ_{psi}. Eles são convertidos em valores de força f_R e f_L na escala compatível com a dos rotores, assim como nos ângulos de referência de servos α_R e α_L (mais detalhes em [Goncalves et al. 2013a]).

Visando a análise do comportamento do VANT, um modelo computacional foi desenvolvido, onde, por meio de equações matemáticas, o desempenho da aeronave é descrito, com base nos valores de entrada informados. Este sistema recebe como entrada os valores de força e referência angular, e a partir destes são calculadas atitude, posição, orientação e velocidade angular do VANT.

A representação do comportamento da aeronave ocorre por meio do cálculo das forças aplicadas sobre o frame *B*. A partir destas as dinâmicas translacionais são descritas, assim como o cálculo da matriz de torque, a derivação dos ângulos de Euler, o cálculo da matriz de inércia e o cálculo da matriz de Coriolis.

Com base nos componentes gerados, são apresentadas as acelerações lineares e angulares do VANT. A partir da integração destas, a representação das velocidades linear e angular são geradas. Integrando esses valores se obtêm a posição da aeronave.

A estrutura de controle descrita foi implementada com base no trabalho de [Chowdhury et al. 2012], o qual descreve as equações de controle aqui aplicadas. O modelo descrito pelos autores visa o controle de um VANT birrotor de configuração tiltrotor. As leis de controle aqui implementadas podem ser substituídas, no futuro, por algoritmos desenvolvidos especificamente para o projeto ProVant.

Aliado ao processo de modelagem matemática e ao desenvolvimento dos algoritmos de controle, a construção da estrutura mecânica do VANT foi realizada.

2.4 CONSTRUÇÃO DA AERONAVE

A montagem mecânica da estrutura do VANT foi iniciada pelo acoplamento do trem de pouso ao corpo principal responsável por receber os conjuntos de propulsão, de forma a estabelecer uma estrutura capaz de suportar os demais componentes da aeronave. Entre o trem de pouso e o corpo principal, uma estrutura intermediária foi instalada a qual tem por objetivo

acoplar os sensores, assim como, as plataformas embarcadas responsáveis pelo gerenciamento do VANT.

A partir desta estrutura inicial os sensores e atuadores foram inseridos no ambiente do VANT, compondo assim a aeronave conforme apresentado na Figura 13

Figura 13 – Protótipo do VANT.

Apesar da concepção da estrutura física do VANT, assim como da implementação dos algoritmos de controle em ambiente de simulação, estes não puderam ser aplicados diretamente à aeronave, pois devido a problemas enfrentados no interfaceamento com os sensores, no acoplamento e alimentação desses a aeronave ainda não esta construída em sua totalidade.

Para que a aplicação da estrutura de software possa ser aplicada às plataformas embarcadas, o processo de interfaceamento com os sensores deve ser finalizado, e novos testes dos algoritmos de controle devem ser realizados. Aliado a isso, o desenvolvimento do sistema da estação base deve ser finalizado, de forma a prover um ambiente propício para essa integração.

Visando o estabelecimento de um ambiente seguro de testes, onde a integridade física da aeronave não fosse colocada em risco, diferentes técnicas de simulação podem ser aplicadas. Dentre elas se têm o *hardware-in-the-loop*, onde se utiliza a plataforma embarcada, utilizada para o controle do processo real, integrada a um modelo computacional, visando a representação do comportamento da aeronave em um ambiente seguro e controlado.

3 REVISÃO BIBLIOGRÁFICA

O projeto de sistemas embarcados descreve um ambiente em que aplicações de tempo real são desenvolvidas e aplicadas a estruturas com um poder de processamento reduzido, e ao controle de processos críticos. Assim, a definição de um ambiente seguro de simulação se torna essencial, visando tanto a concepção quando a validação destes sistemas.

Buscando auxiliar o processo de modelagem e desenvolvimento dos sistemas embarcados aplicados ao contexto do VANT, e visando prover um ambiente seguro e confiável para sua concepção, é possível utilizar a técnica de emulação em HIL. Ela têm por objetivo a união entre o ambiente simulado e a plataforma embarcada real, de forma a abstrair os riscos físicos da modelagem, e a aplicação dos sistemas diretamente na estrutura final da aplicação [Louall et al. 2011].

As aplicações desenvolvidas nestes ambientes têm por características a iteração direta com o ambiente em que estão inseridas, bem como o atendimento às restrições temporais impostas. Dessa forma, a preocupação fundamental dessas aplicações se concentra em organizar o conjunto de tarefas de forma que sejam concluídas dentro do tempo disponível. No entanto, o atendimento dessas restrições temporais é uma tarefa complexa, e exige um esforço adicional dos desenvolvedores na modelagem de técnicas que permitam o cumprimento dos prazos.

A técnica de computação imprecisa, dos *Anytime Algorithms*, é uma abordagem alternativa, que permite introduzir flexibilidade ao sistema pela utilização de diferentes níveis de precisão e reduzir os níveis de utilização de CPU mantendo a estabilidade do sistema [Mangharam e Saba 2011].

Para guiar o processo de desenvolvimento dos sistemas embarcados, seja na concepção do ambiente HIL ou na aplicação dos AA, metodologias podem ser aplicadas a estes ambientes, de forma a permitir aos projetistas uma maior cobertura de todos os aspectos de projeto.

Nesse contexto, são descritos neste capítulo a técnica de simulação em HIL, os *Anytime Algorithms*, e metodologias aplicadas ao desenvolvimento de CPS, apresentando os conceitos aplicados ao ambiente do VANT.

3.1 *HARDWARE-IN-THE-LOOP*

O HIL é descrito como a técnica de emulação caracterizada pela integração entre a plataforma de controle real aliada a um modelo computacional (Figura 14), e por meio dessa estrutura o processo real pode ser computaci-

onalmente descrito e analisado. Essa técnica é aplicada ao desenvolvimento, teste e validação de sistemas de tempo real embarcados complexos, proporcionando uma plataforma efetiva permitindo a adição de complexidade, com a segurança da plataforma de testes [Louall et al. 2011].

Figura 14 – Estrutura HIL.

Os sinais de entrada e de saída apresentam seus valores de acordo com o comportamento do processo real. Dessa forma, o sistema embarcado pode ser submetido a diferentes cenários [Shixianjun et al. 2006].

Por meio de seu uso, os pesquisadores podem verificar efetivamente os aspectos comportamentais e temporais do sistema desenvolvido. Portanto, a utilização desse ambiente de simulação vem sendo intensificada na verificação do desempenho de estratégias de controle. Ademais, ajustes e modificações podem ser realizados no sistema antes da sua aplicação definitiva no ambiente real, prevendo assim a ocorrência de acidentes, e contribuindo para minimização dos riscos [Cai et al. 2008a].

O HIL também inclui a representação dos sensores e atuadores. Os sinais gerados por eles estabelecem uma interface entre a planta emulada e o sistema embarcado. Os dados extraídos da planta emulada são encaminhados como leitura para o sistema embarcado. A partir desta troca de informações, os subsistemas são alimentados e, de acordo com os testes propostos, os comportamentos tanto do sistema embarcado quanto do modelo computacional podem ser verificados.

Comparado com simulações numéricas e físicas, o HIL apresenta diversas vantagens. Inicialmente, os resultados das simulações HIL são mais confiáveis que os das simulações numéricas pela proximidade do ambiente real em que estão inseridos e, a partir da sua utilização, pode-se economizar recursos e tempo dos projetistas visto que os algoritmos de controle são testados e ajustados diretamente na plataforma alvo. Além disso, essa técnica provê uma plataforma eficiente onde os projetistas e pesquisadores podem corrigir ou melhorar seus projetos originais [Shixianjun et al. 2006].

No desenvolvimento de um simulador HIL é necessário a implementação da planta de simulação de tempo real que descreve, por meio de represen-

tação matemática, todas as dinâmicas incluídas na planta do sistema [Louall et al. 2011].

Desta forma, o HIL tem sido adotado com objetivo de validar de forma conjunta o hardware e o software sob diferentes condições de ambiente [Jung e Tsiotras 2007].

No contexto dos VANTs, o HIL têm sido aplicado com diferentes finalidades, que vão da validação dos algoritmos de controle à aplicação de softwares de simulação de voo para verificação de comportamento. A aplicação do HIL no ambiente de desenvolvimento dos VANTs visa facilitar o seu processo de concepção. A seguir são descritos trabalhos que fazem uso do HIL com VANTs.

3.1.1 [Cai et al. 2008a] *Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters*

Em [Cai et al. 2008a] os autores descrevem a utilização do HIL no processo de desenvolvimento de helicópteros de pequena escala.

Um ambiente é proposto visando à análise dos algoritmos de controle desenvolvidos. Os VANTs projetados foram criados a partir de dois modelos comerciais, onde a partir de algumas modificações em sua estrutura deram origem aos veículos autônomos HeLion e SheLion.

O modelo HIL desenvolvido descreve a interação entre os VANTs e a arquitetura de software proposta, aliado a um modelo responsável pela simulação das dinâmicas do helicóptero. Assim, esta plataforma é composta por quatro módulos: o módulo de hardware embarcado; o módulo de controle de voo; o módulo da estação de solo; e o módulo de software.

O módulo de hardware é composto por três componentes: a plataforma embarcada; os atuadores responsáveis pelo movimento do helicóptero; e um modem wireless. O controle de voo é realizado na plataforma embarcada, dedicada somente a esta função. Um segundo hardware compõe a estrutura da aeronave, de forma a realizar as demais tarefas do sistema. O segundo componente de hardware é composto pelos atuadores responsáveis pela movimentação do VANT. O modem wireless corresponde ao terceiro componente incluído na estrutura do HIL, ele provê um canal de comunicação sem fio *full-duplex* para a troca de informações.

O controle automático de voo é implementado no segundo módulo. Sua estrutura consiste de um modelo hierárquico composto por três laços: o mais interno, responsável por estabilizar as dinâmicas do helicóptero; um intermediário que controla a posição do helicóptero; e o mais externo que realiza a navegação do VANT de acordo com a trajetória gerada na estação

base.

O módulo da estação de solo tem como função principal o monitoramento do VANT. No ambiente HIL, o desempenho do sistema como um todo é avaliado de acordo com os dados recebidos da aeronave. Outra característica importante relacionada à estação base é a capacidade do envio de comandos e tarefas em tempo real diretamente para o VANT, caso haja necessidade.

A integração dos módulos descritos é de responsabilidade da arquitetura de software, dividida em duas partes: o software embarcado no VANT e o software da estação base. O sistema embarcado é descrito como um cenário multitarefas (do inglês *multi-thread*), onde a estrutura é definida por uma tarefa principal e cinco tarefas de sistema descritas como: controle de voo do helicóptero; aquisição de dados; controle dos atuadores; comunicação com a estação base; e o armazenamento de dados. Estas são gerenciadas pela *thread* principal, responsável pela ativação da execução das demais tarefas.

A software da estação de solo é implementado essencialmente por duas camadas de primeiro e segundo plano. A transferência de dados é executada em segundo plano, onde se estabelece um canal de comunicação com o sistema embarcado. Já a camada de primeiro plano é responsável por apresentar ao usuário os dados referentes ao voo. Assim, diferentes visões da aeronave são apresentadas, bem como leituras referentes ao voo executado.

Com base no ambiente HIL desenvolvido, três casos de testes foram apresentados: a simulação dos movimentos básicos de voo; a simulação de um voo completo; e a execução de voos com múltiplos VANTs.

A simulação de movimentos básicos tem por objetivo analisar o comportamento do VANT na realização de movimentos essenciais para realização dos voos autônomos. Garantir estabilidade e desempenho confiável destes movimentos no VANT é essencial, possibilitando a execução de operações que exijam mais funcionalidades.

Nos testes de voo o VANT é submetido a uma série de passos que descrevem seu comportamento no ar. Incluindo: decolagem automática (*automatic takeoff*); deslizamento (*slithering*); movimento de guinada (*head turning*); pirueta (*pirouetting*); giro vertical (*vertically wheeling*); espiral descendente (*downward spiraling*); e o pouso automático (*automatic landing*).

Nas simulações de voos em formação realizada pelos autores, por razões de segurança, a estação base foi estabelecida como líder virtual e utilizada um VANT real como seguidor. Estabelecendo assim que a aeronave seguidora mantenha sete metros de distância de seu líder.

Os autores propuseram um ambiente HIL aplicado a VANTs helicópteros de pequena escala. Os resultados obtidos em simulação nos três ambientes foram comparados com os dados apresentados em voos reais e desta forma se

pode verificar que o sistema HIL proposto é capaz de prever com precisão e eficiência simulações reais de voo, incluindo riscos potenciais de acidentes.

3.1.2 [Jung e Tsiotras 2007] *Modeling and Hardware-in-the-Loop Simulation for a Small Unmanned Aerial Vehicle*

A validação de algoritmos de controle também é realizada por meio da utilização de um ambiente HIL. No instituto tecnológico da Geórgia uma abordagem de HIL é descrita por [Jung e Tsiotras 2007], onde um ambiente de simulação para validação de sistemas de controle aplicados a VANTs é apresentado.

O modelo busca recriar as condições do ambiente real em um modelo computacional, sendo este baseado no Matlab/Simulink. Um modelo não linear da aeronave de seis graus de liberdade com uma aproximação linear das forças e dos momentos é utilizado para representar o comportamento dinâmico realista da aeronave.

Quatro plataformas computacionais independentes são utilizadas no ambiente HIL: o simulador de seis graus de liberdade[1]; o computador responsável pela visualização do voo; o microcontrolador de piloto automático; e o computador responsável pela estação de solo.

O modelo não linear da aeronave foi desenvolvido no ambiente Matlab/Simulink. As funções desempenhadas pelo sistema de piloto automático em simulação são idênticas às desempenhadas em voos reais, descrevendo os comandos aplicados aos atuadores. Estas informações são enviadas ao modelo de forma a guiá-lo de acordo com os sinais de controle.

O ambiente de simulação se aproxima muito do modelo real, dada a aplicação de características como um modelo de gravidade e um campo magnético. As saídas do simulador são processadas de forma a emular o comportamento dos sensores, representando características como latência e ruídos de medição. Após a digitalização das leituras, os dados são transmitidos ao piloto automático por meio de comunicação serial.

Um transmissor de rádio controle é conectado ao modelo para permitir a atuação de um piloto real, caso haja necessidade. Os comandos descritos por ele podem substituir o piloto automático a qualquer momento. Neste caso a simulação realiza uma manobra em malha aberta e o avião pode ser conduzido de forma manual validando as características dinâmicas do modelo da aeronave.

A aeronave simulada é visualizada por meio do uso do FlighGear, uma

[1] O modelo de seis graus de liberdade descreve a posição da aeronave (representada pelas variáveis X, Y e Z), e a sua orientação (descrita por meio dos ângulos ϕ, θ e ψ).

ferramenta de simulação de voo de código aberto. Embora este forneça total simulação tanto do ambiente quanto das dinâmicas de voo, os autores optaram pela utilização do simulador aliado ao seu modelo já desenvolvido. Desta forma o FlightGear recebe os dados a uma taxa fixa de transmissão, e atualiza o comportamento da aeronave.

O modelo computacional combinado à visualização das dinâmicas da aeronave por meio do FlightGear permite que a ferramenta de simulação desenvolvida possa substituir uma experiência de voo real, economizando tempo e esforço. Assim como, o comportamento em malha aberta pode ser testado e validado por um piloto remoto por meio de um dispositivo de entrada amigável, e o desempenho do piloto automático também pode ser verificado pela realização de experimentos virtuais antes da sua aplicação em um voo real.

Neste contexto um ambiente HIL proposto pelos autores permite validar o hardware e o software desenvolvidos. A utilização desta técnica se torna indispensável nestes ambientes, possibilitando a simulação e testes de voos e dando suporte ao processo de projeto de sistemas de controle aplicados ao ambiente dos VANTs com menor custo e esforço.

Visto que o ambiente HIL pode ser desenvolvido com diferentes configurações, os trabalhos descritos apresentam sua aplicação na validação de algoritmos de controle de voo autônomo. No entanto, devido a fato destes não utilizarem uma plataforma embarcada similar a aplicada no ProVant, e da configuração das aeronaves desenvolvidas também ser diferente, o desenvolvimento de um novo modelo de HIL foi necessário neste projeto, de forma a representar as dinâmicas da aeronave desenvolvida e integrar com a plataforma embarcada selecionada, conforme descrito na Seção 4.3

Considerando a criticidade dos sistemas embarcados aplicados aos VANTs, juntamente com hardware utilizado que muitas vezes apresenta recursos limitados, o uso de técnicas auxiliares ao ambiente do HIL é praticado a fim de melhor utilização destes. Nesta estrutura, técnicas são propostas visando maximizar o desempenho das aplicações desenvolvidas, como é o caso da técnica *Anytime Algorithms*, apresentada como uma abordagem voltada ao desenvolvimento de sistemas embarcados críticos.

3.2 ANYTIME ALGORITHMS

Diferentemente das abordagens usuais de tempo real, onde se busca apresentar soluções dentro do tempo disponível, pesquisadores propuseram uma abordagem na qual o melhor resultado possível é apresentado dentro do tempo que lhes é disponibilizado. Nesse modelo, a técnica proposta visa sa-

crificar a qualidade dos resultados com intuito de cumprir os prazos exigidos. Por esse motivo, essa abordagem é referenciada como computação imprecisa, de forma que permita a flexibilização das execuções da aplicação de tempo real [Liu et al. 1994].

Dentro deste cenário existem os *Anytime Algorithms* [Dean e Boddy 1988], algoritmos cuja execução pode ser interrompida a qualquer instante e a precisão de seus resultados depende do tempo de execução.

Esses algoritmos são caracterizados principalmente pela capacidade de suspensão e retomada sem grandes esforços computacionais, assim como, por apresentarem sempre uma resposta independente do tempo em que o algoritmo é encerrado. Nesse modelo, a qualidade da resposta apresentada é incrementada de acordo com o tempo dispensado a mesma [Mangharam e Saba 2011].

Sua implementação é tradicionalmente considerada sob duas variações, sendo estas, a aplicação dos algoritmos interruptíveis (do inglês *Interruptible Algorithms*), ou pelo uso dos algoritmos de tempo de contato (do inglês *Contact-time Algorithms*).

Algoritmos interruptíveis podem ser interrompidos a qualquer instante, sendo assim o melhor resultado obtido até o ponto em que este foi processado é apresentado. Devido a essas características, esses algoritmos são recomendados para utilização em ambientes onde os resultados são solicitados sob demanda.

Os algoritmos de tempo de contato são especificados em tempo de execução, onde se define a melhor estratégia do algoritmo de forma a maximizar a qualidade do resultado gerado dentro do tempo de processamento disponível. Essa abordagem se torna mais intuitiva, pois usa de estruturas de dados e de controle simplificadas em seu desenvolvimento, o que facilita os processos de codificação e manutenção.

Os *Anytime Algorithms* têm sido aplicados ao desenvolvimento de sistemas embarcados, os trabalhos listados a seguir descrevem seu uso em diferentes contextos.

3.2.1 [Quagli et al. 2009] *Designing real-time embedded controllers using the anytime computing paradigm*

Em [Quagli et al. 2009] os autores descrevem uma abordagem de controle *anytime*, aplicado ao modelo de um helicóptero. Para a implementação das leis de controle, elas foram divididas em segmentos, onde de acordo com a execução destes, um refinamento no resultado é realizado. Desta forma, quanto maior o refinamento mais agressivas são as respostas apresentadas.

Nesta estratégia o primeiro segmento é sempre executado, de forma a garantir sempre uma solução, enquanto os demais são ativados apenas quando há tempo de computação suficiente.

Para o desenvolvimento do sistema os autores propõem uma metodologia que visa atender aos seguintes requisitos: o desenvolvimento de uma sequência de objetivos associados a um desempenho crescente; o projeto de um controlador iterativo baseado em segmentos subsequentes que atinja aos objetivos; e a modelagem da interface entre o controlador e a plataforma alvo.

Considerando o modelo do helicóptero, é definida uma malha de controle em três níveis: um nível básico que tem por objetivo a estabilidade assintótica do modelo (nível 1); um segundo nível responsável pela rejeição da perturbação (nível 2); e um terceiro nível que descreve uma minimização na qual é possível considerar as exigências anteriores de uma única vez (nível 3).

Com base nos tempos de computação das tarefas e no processo de escalonamento, os autores propõem a construção de um controlador *anytime* de acordo com os seguintes critérios: a execução do nível 1 do controlador deve sempre ser garantida, sendo baseada em seu pior caso de execução; havendo tempo computacional suficiente, passos adicionais podem ser executados visando atingir os níveis 2 e 3.

Para viabilizar essa abordagem, um processo de desenvolvimento foi proposto, baseado nas seguintes etapas: a apresentação e caracterização do ambiente de execução; o projeto do controlador de forma que os níveis descritos sejam incrementais e obtidos pela execução de diferentes sub-rotinas; e a escolha de uma política de compartilhamento que permita a execução de qualquer uma das rotinas garantindo a estabilidade global do sistema.

O desenvolvimento do modelo foi realizado por meio do uso da ferramenta Matlab/Simulink TrueTime, onde um conjunto formado por duas tarefas foi simulado. A primeira implementa o controle *anytime*, enquanto a segunda, que possui prioridade maior, implementa o mecanismo de inferência responsável pelo agendamento da tarefa.

A partir da concepção da abordagem de controle *anytime*, esta foi aplicada ao ambiente de um helicóptero com dois graus de liberdade. Ele foi desenvolvido utilizando a plataforma embarcada PXI, que possui um sistema operacional de tempo real embarcado. Para monitoramento do sistema, a plataforma embarcada foi integrada ao ambiente LabVIEW.

O LabVIEW realiza o rastreamento das tarefas agendadas no sistema embarcado e de acordo com os dados apresentados, o comportamento *anytime* é retratado pela tarefa de controle.

Neste ambiente testes foram realizados, onde a aeronave era iniciada em uma posição e sofria uma perturbação gerada por uma força externa. A

partir desta se verificava a capacidade do sistema em restaurar o equilíbrio anterior por meio do controlador proposto.

Neste contexto uma proposta de controle automático em conformidade com as características *anytime* foi apresentada, e a aplicação em um ambiente de testes foi realizada. Dada à eficácia apresentada, os autores descrevem a aplicação desta técnica como alternativa viável visando um melhor aproveitamento dos recursos computacionais embarcados.

Além da aplicação direta aos algoritmos de controle, os AA também são utilizados no planejamento de trajetórias, conforme descrito a seguir.

3.2.2 [Narayanan et al. 2012] *Anytime Safe Interval Path Planning for dynamic environments*

Um planejador de trajetórias que aplica os *Anytime Algorithms* é descrito em [Narayanan et al. 2012], onde este visa estabelecer uma rota segura para a aeronave. A proposta desenvolvida é aplicada descrevendo o planejamento de trajetória de um VANT em um ambiente que apresenta obstáculos dinâmicos.

A aplicação do planejador *Anytime* tem por objetivo apresentar soluções no menor tempo possível, e havendo tempo disponível o resultado fornecido é incrementado. Em ambientes dinâmicos, se o planejador requer muito tempo para retornar um novo caminho, uma colisão pode ocorrer com um obstáculo em movimento.

O modelo proposto realiza o planejamento *Safe Interval Path Planning* (SIPP), sendo este realizado por meio do uso do algoritmo A*. A característica *Anytime* foi adicionada por meio de uma extensão do SIPP, de forma a permitir a sua execução *anytime*. Esta é o resultado da combinação do *Anytime Repairing A** (ARA*), onde a capacidade de execução de pesquisa do A* é ponderada, com o SIPP.

A abordagem proposta assume que há um sistema externo responsável pelo monitoramento dos obstáculos dinâmicos do ambiente, prevendo suas trajetórias futuras e listando estes dados. Por meio desta lista descrevendo os dados dos obstáculos ordenados pelo mais próximo, o planejador prevê a sua movimentação em um futuro próximo e define a rota mais segura para a aeronave.

O ARA* é iniciado pela execução de uma pesquisa de A* com o critério de parada mais alto pré-estabelecido, desta forma é produzido rapidamente uma solução inicial. Caso haja tempo suficiente, o ARA* reduz o valor do critério de parada, de forma a incrementar o resultado, enquanto o A* reutiliza as pesquisas anteriores na busca por um novo resultado. Havendo

tempo suficiente o critério de parada do ARA* decrementará até o valor um (1), retornando assim a solução ótima.

Visando analisar o comportamento do planejador, diferentes cenários foram propostos variando a quantidade de obstáculos dinâmicos envolvidos. Assim, foi possível verificar que o modelo proposto é capaz de produzir respostas em tempo real e de fornecer soluções até um horizonte de tempo de 15 segundos, mesmo quando aplicado a grandes ambientes com muitos obstáculos dinâmicos.

A aplicação do *anytime* se mostrou eficaz no planejamento de trajetórias, seja para a localização de soluções de forma rápida, ou para o incremento destes resultados caso haja tempo de processamento disponível.

Conforme apresentado, os AA têm sido aplicados com diferentes propósitos no contexto do VANT, no entanto devido à diferença de configuração das aeronaves, a diferença de plataformas embarcadas e da diferença das estratégias de controle, uma abordagem *Anytime* diferente das apresentadas nestes trabalhos foi aplicada no ProVant.

Aliado ao uso das técnicas de HIL e *Anytime Algorithms*, a aplicação de metodologias de desenvolvimento ao ambiente de sistemas embarcados também é realizada, visando guiar os engenheiros no processo de modelagem. Essas buscam organizar o trabalho dos projetistas e reduzir tempo e custo de desenvolvimento dos produtos. A seguir são apresentados alguns trabalhos que descrevem propostas de metodologia aplicadas ao desenvolvimento de sistemas embarcados.

3.3 METODOLOGIAS DE DESENVOLVIMENTO APLICADAS A SISTEMAS EMBARCADOS

Nesta seção são apresentadas algumas metodologias propostas por diferentes autores com intuito de guiar o processo de desenvolvimento dos sistemas e auxiliar os projetistas.

3.3.1 [Becker et al. 2010] *Development Process for Critical Embedded Systems*

Em [Becker et al. 2010] os autores propõem uma metodologia aplicada ao desenvolvimento de sistemas embarcados críticos. Esta abordagem busca, por meio de técnicas de verificação de modelos, atender as necessidades de projeto.

Através da engenharia baseada em modelos (do inglês *Model Driven*

Engineering - MDE) [Schmidt 2006] se busca realizar a verificação de modelos do sistema proposto, sendo estes desenvolvidos na linguagem *Architecture Analysis & Design Language* (AADL) [Feiler et al. 2006], de forma a validar o atendimento aos requisitos não funcionais.

A metodologia de projeto descreve que a arquitetura do sistema é resultado de diversas etapas de verificação, de modo a assegurar sua exatidão. Para alcançar esse objetivo uma sequência de transformações de modelo é realizada, sendo estas iniciadas pelo modelo AADL e finalizadas com um modelo de autômato passível de verificação.

Neste contexto, sete passos de desenvolvimento são propostos. No entanto apesar de sequencialmente descritos, eles não necessitam da sua execução sequencial, possibilitando a atualização ou a reexecução de alguns destes no decorrer do processo. Os passos são descritos como:

1. Definição de requisitos;
2. Modelagem funcional;
3. Descrição do ambiente;
4. Modelagem da arquitetura de software;
5. Mapeamento software e hardware;
6. Refinamento das propriedades de tempo real;
7. Verificação.

O passo inicial descreve o levantamento e a especificação dos requisitos funcionais e não funcionais, sendo seguido pelo desenvolvimento do modelo funcional do sistema (passo dois). Para a concepção dos modelos, ferramentas como Scade/Lustre e Matlab/Simulink podem ser utilizadas.

A metodologia proposta é realmente iniciada no terceiro passo, onde a construção do modelo AADL é realizada. Neste são especificados os dispositivos que interagem com o sistema proposto. Dessa forma, o ambiente de contato do sistema computacional a ser projetado é descrito.

No quarto passo, a modelagem da arquitetura de software e hardware é desenvolvida. Esse é um dos passos mais importantes, dado o nível de detalhamento agregado ao modelo, ele é dividido em duas partes: a primeira compreende a modelagem e verificação da arquitetura de software; e a segunda descreve a modelagem da arquitetura de hardware. Um modelo AADL que descreve as propriedades básicas já verificadas do sistema e uma arquitetura de hardware potencialmente capaz de executar o modelo de software projetado é gerado como resultado desta etapa.

O processo de projeto visa detalhar o modelo AADL, que pode resultar na derivação de vários subcomponentes. Assim, processos como o refinamento de arquitetura e verificação do modelo são realizados nesta etapa.

O mapeamento dos componentes de software e hardware é desenvolvido no quinto passo. Nele, as tarefas são associadas ao seu processador específico, permitindo assim, a verificação das propriedades do sistema.

O sexto passo é o refinamento das propriedades de tempo real do modelo AADL. Ele é realizado com base nas informações de tempo precisas obtidas no processo de simulação realizado no passo anterior. Além disto, por meio de análises, a estimação do pior caso de execução das tarefas pode ser realizada.

O ciclo de desenvolvimento é concluído com a realização da verificação final (sétimo passo), onde o modelo AADL é inserido como entrada do sistema e atualizado com base nos princípios do MDE, assim como a corretude deste também é verificada. A partir deste processo torna-se possível realizar a geração automática do código da aplicação.

O processo de verificação é realizado de forma a confirmar as propriedades de lógica linear temporal (do inglês *Linear Temporal Logic* - LTL) [Chaki et al. 2004] do modelo desenvolvido. Além disto, outras propriedades podem ser verificadas, como escalonabilidade, estouro de buffer e propriedades definidas pelo usuário.

Visando a realização das verificações, o modelo AADL é transformado em um modelo Fiacre [Berthomieu et al. 2008]. Este então sofre uma segunda transformação gerando um modelo TTS [Henzinger et al. 1992] e a partir deste, um autômato com as propriedades do sistema é gerado. Estas então são verificadas por meio do autômato, de forma a diagnosticar o sistema analisado.

Os autores propuseram uma abordagem de verificação e um conjunto de ferramentas que visam auxiliar na concepção de sistemas críticos, por meio de um modelo AADL. Assim, a partir da concepção e verificação dos modelos ainda se torna possível à geração automática de código para uma plataforma alvo a partir deste.

Uma segunda proposta é descrita a seguir, onde os autores descrevem uma abordagem voltada ao desenvolvimento de VANTs.

3.3.2 [Cai et al. 2008b] *Systematic design methodology and construction of UAV helicopters*

Uma proposta voltada ao desenvolvimento de VANTs do tipo helicóptero é descrita em [Cai et al. 2008b]. Ela foi criada com base na experiência

adquirida pelos autores no desenvolvimento de projetos anteriores, onde foi construído o VANT denominado HeLion. Neste trabalho a metodologia é aplicada ao desenvolvimento de uma segunda aeronave denominada SheLion.

O método proposto é baseado em quatro passos, sendo estes: o projeto computacional do modelo da aeronave; a seleção dos componentes de hardware; a integração destes componentes; e os testes de avaliação em solo e em voo.

O projeto computacional descreve a construção de um modelo da aeronave, sendo esta etapa suportada por uma ferramenta de modelagem. Para o projeto proposto os autores usaram o SolidWorks, ferramenta que permite a análise de diferentes características como, funcionalidade e cálculo do centro de gravidade, além de proporcionar um modelo em três dimensões.

Com base no modelo, os componentes de hardware são especificados. Desta forma, os sensores, os atuadores e a plataforma embarcada são selecionados. A aeronave desenvolvida pelos autores tem como base um modelo radio controlado comercial onde, a partir da sua estrutura, foram especificados os demais sensores necessários para realização de voos autônomos. O módulo radio controlado foi mantido, pois caso haja a ocorrência de algum fato adverso o modo manual pode ser habilitado.

A partir da seleção dos componentes de hardware a integração sistemática destes componentes é realizada. Nesta fase os componentes selecionados são acoplados a estrutura do VANT. Buscando o melhor posicionamento de cada um destes, o layout de acoplamento é desenvolvido previamente. Aliado ao projeto do layout, questões relacionadas à alimentação, vibração e à interferência eletromagnética também são verificadas.

A construção da aeronave ocorre com base no modelo computacional. Desta forma, se obtêm o modelo físico do VANT. A partir desta estrutura, uma série de testes tanto em solo quanto em voo é realizada, visando avaliar desempenho e confiabilidade da aeronave (quarto passo).

Uma terceira proposta de metodologia é descrita a seguir. Ela visa guiar o processo de desenvolvimento de sistemas ciber-físicos.

3.3.3 [Jensen et al. 2011] *A model-based design methodology for cyber-physical systems*

O desenvolvimento de *Cyber-Physical Systems* é um processo complexo que deve ser planejado cuidadosamente visando o seu sucesso. Desta forma, metodologias baseadas em projeto baseado em modelos (do inglês *Model-Based Design - MBD*), podem ser aplicadas em todas as fases do projeto.

Em [Jensen et al. 2011] os autores apresentam uma proposta que visa guiar o processo de desenvolvimento de CPS. A proposta sugere a utilização de dez passos fundamentais de desenvolvimento. No entanto, apesar da sua enumeração, os passos não são sequenciais, mas codependentes, facilitando assim a coevolução do modelo. Esses passos são:

1. Definição do problema;
2. Modelagem dos processos físicos;
3. Caracterização do problema;
4. Obtenção dos algoritmos de controle;
5. Seleção dos modelos para computação;
6. Especificação do hardware;
7. Simulação do sistema;
8. Construção;
9. Síntese do software;
10. Verificação, validação e testes.

O primeiro passo busca apresentar uma clara descrição do problema abordado. Este procedimento pode ser realizado em linguagem simples, sem a necessidade do uso de termos matemáticos ou técnicos. No entanto, deve representar efetivamente os requisitos de projeto.

A seguir a modelagem do sistema físico é descrita (segundo passo), onde os sistemas físicos que serão controlados devem ser modelados. Os sistemas são usualmente representados na forma de equações diferenciais ou funções de transferência. A representação pode ser iniciada por uma versão simplificada, a qual será refinada durante o processo de desenvolvimento.

No terceiro passo, a caracterização do problema, os parâmetros fixos e as variáveis de controle do sistema são isolados e ajustados, identificando os valores de referência que representam os processos físicos, tais como, espaço de configuração, limitações de segurança, conjuntos de entradas e saídas, pontos de saturação e comportamento do modelo. Esse passo permite a compreensão da integração do processo físico com o processo computacional.

Com base na descrição do processo físico, o quarto passo compreende a obtenção dos algoritmos de controle que serão executados na plataforma embarcada. Nesta etapa a descrição do problema é utilizada para definir os requisitos não funcionais aplicados. O refinamento deste passo é necessário

após a especificação do modelo computacional e da especificação de hardware, para que se permita uma análise do impacto das latências e jitters introduzidos seja pelo modelo de computação assíncrona, saturação, ou outros fatores não lineares estabelecidos pelo hardware utilizado.

A próxima fase do projeto (passo cinco), consiste na seleção dos modelos de computação, onde o modelo computacional é definido como um conjunto de instruções e regras responsáveis por gerenciar as iterações, a comunicação e o controle de fluxo dos componentes computacionais. Com base nestes, a análise de determinismo dos tempos de execução, da acessibilidade dos estados, do uso de memória e da latência do sistema são facilitadas.

Na especificação de hardware (sexto passo), os projetistas definem a arquitetura de hardware apropriada, sendo que esta deve ser capaz de: oferecer suporte ao ambiente em que esta inserida; interagir com processos físicos; e implementar os algoritmos de controle. A definição da plataforma embarcada é diretamente dependente da análise das restrições impostas pelo sistema e da verificação de como o software desenvolvido interage com a arquitetura de hardware especificada.

A simulação do sistema representa o passo sete. Para dar suporte a este procedimento, ferramentas de simulação são utilizadas visando analisar o comportamento do sistema desenvolvido sob as mais diversas condições em que este pode ser submetido. O Projeto baseado em plataforma (do inglês *Platform-based design*)[2] pode ser aplicado nesta etapa, a fim de abstrair a lógica de aplicação a partir das dinâmicas do sistema, de modo a facilitar o processo de geração de código.

Consolidado o projeto do CPS, no oitavo passo o modelo é concebido com base nas descrições, exceções e limitações impostas.

O passo nove compreende o processo de transformação dos modelos desenvolvidos em aplicação. Este processo normalmente é realizado de forma automática por meio das ferramentas de codificação. No entanto, a transcrição manual também pode ser realizada pela equipe de desenvolvimento, seja pela ausência dessa funcionalidade na ferramenta, seja por algum outro motivo que inviabilize o processo automatizado.

A etapa final do processo de planejamento tem como foco a realização da verificação, validação e testes. Nessa fase o sistema é ajustado de forma que todos os componentes e subsistemas possam ser testados de forma individual e independente.

Por meio dos passos descritos os autores buscam cobrir todas as fases de projeto e concepção destes sistemas, de forma a proporcionar uma me-

[2] A técnica do projeto baseado em plataforma permite que uma única plataforma base possa servir para uma linha inteira de projetos, com a modificação de apenas algumas seções configuráveis, intencionalmente dispostas para este fim.

lhor organização tanto do andamento quanto da documentação gerada pelo projeto.

Analisando as metodologias descritas e considerando as características ciber-físicas do VANT e a abrangência do ProVant, que aborda desde a especificação da aeronave até a modelagem dos algoritmos de controle, se optou por utilizar a metodologia descrita por [Jensen et al. 2011]. Essa foi definida devido ao fato de apresentar uma maior cobertura de todos os aspectos envolvidos no desenvolvimento do VANT.

Considerando o projeto seguro de sistemas embarcados, tem-se a utilização das técnicas de HIL e dos AA como facilitadores deste processo, assim como, a aplicação das metodologias de desenvolvimento que buscam guiar este processo atendendo a todos os seus aspectos. O capítulo a seguir descreve os aspectos de projeto seguro do sistema, apresentando desde a aplicação da metodologia no contexto do VANT, a modelagem funcional, o projeto de HIL e a aplicação dos AA.

4 PROJETO SEGURO DO VANT

O processo de desenvolvimento dos CPS é uma tarefa delicada, visto que estes estabelecem uma ponte entre o mundo computacional e o ambiente real [Lee 2008]. Desta forma, seu projeto exige que uma série de fases sejam contempladas, tais como análise do ambiente, especificação de processos, definição de variáveis e parâmetros, entre outras.

O projeto seguro de sistemas prevê a especificação, a organização e o gerenciamento de todos os artefatos produzidos neste processo. Esse descreve a especificação do sistema, o desenvolvimento do seu modelo funcional, a especificação de um ambiente de emulação e testes, e a análise do seu comportamento.

Visando garantir e organizar essa sequência de ações, metodologias são propostas de forma a facilitar o processo de desenvolvimento, assim como, ferramentas são apresentadas buscando auxiliar na concepção dos sistemas e na garantia de segurança do projeto.

Este capítulo descreve a aplicação de uma metodologia no contexto do ProVant, o desenvolvimento da modelagem funcional do sistema representada por meio de um diagrama de blocos construído no Simulink, o estabelecimento de um ambiente de emulação em HIL e a especificação de alguns casos de teste.

A metodologia descrita por [Jensen et al. 2011] foi aplicada neste projeto com objetivo de guiar o processo de desenvolvimento do VANT. Detalhes referentes à modelagem e da aplicação da metodologia selecionada são detalhados na próxima seção.

4.1 METODOLOGIA APLICADA NO PROJETO PROVANT

Com base em [Jensen et al. 2011] o projeto do VANT foi iniciado com a realização da descrição do sistema, apresentada no Capítulo 2. A partir destas especificações, os sistemas físicos foram descritos de forma similar ao processo descrito no passo dois da metodologia.

A modelagem do sistema consiste na criação de sua representação computacional, que descreve os controles de estabilidade e seguimento de trajetória, os sistemas de monitoramento de segurança, o interfaceamento com os sensores e o subsistema de gerenciamento dos processos. Inicialmente uma versão simplificada foi desenvolvida, onde esta sofreu refinamentos durante a evolução do projeto.

A definição do conjunto de variáveis necessárias para a modelagem

funcional foi especificada a partir da concepção do modelo computacional.

Os algoritmos de controle foram derivados com base em [Chowdhury et al. 2012], onde este os subdivide em dois níveis, controle de seguimento de trajetória e de estabilidade, os quais são detalhados na seção 2.3.

Um modelo computacional foi concebido visando representar o comportamento do VANT e a estrutura de controle que representa o passo cinco da metodologia. O detalhamento deste processo é descrito na Seção 4.2.1 onde, por meio das simulações realizadas, análises comportamentais entre outros aspectos puderam ser verificados.

A arquitetura de hardware empregada foi especificada por meio do modelo computacional (passo seis da metodologia). Detalhes técnicos desta arquitetura são apresentados na Seção 2.1.

O processo de simulação ocorreu em um ambiente HIL, que permite a integração da plataforma alvo ao modelo computacional. Diferentes cenários foram propostos visando analisar o comportamento, tanto do sistema embarcado, quanto do próprio VANT. O detalhamento tanto do ambiente de simulação quanto dos cenários descritos são apresentados na Seção 4.3.

O processo de síntese de software foi realizado de forma a embarcar os algoritmos de controle, inicialmente desenvolvidos no Simulink, na plataforma embarcada. Esta transição ocorreu de forma manual, pois o código gerado automaticamente pelo Simulink não se mostrou flexível às modificações que se faziam necessárias, tais como ajuste de período, escalonamento de tarefas entre outros. O processo de transcrição é descrito no na Seção 4.3.

Verificação, validação e testes são processos que estão presentes no decorrer de todas as etapas. A cada modelo planejado, algoritmo desenvolvido, ou mesmo no processo de construção, estes processos são extremamente necessários de forma a garantir fatores como integridade, robustez e segurança.

A seguir a descrição do sistema e o modelo de blocos desenvolvido são descritos.

4.2 DESCRIÇÃO DO SISTEMA

A modelagem do sistema de software do VANT descreve a representação de sua arquitetura, responsável pela execução de todos os subsistemas. Desta forma, é necessário que sejam descritos desde os sistemas de baixo nível, responsáveis pelo interfaceamento com os sensores e atuadores, até os de alto nível, que descrevem os algoritmos de controle de voo.

A estrutura de software do VANT é organizada em três subsistemas: o interfaceamento com os sensores; o interfaceamento com os atuadores; e o controle.

O primeiro subsistema é responsável pelo interfaceamento e processamento de dados, onde são reunidas e processadas as informações dos sensores. Este também estabelece comunicação com os atuadores visando à coleta de dados.

Posição e velocidade linear são fornecidas pelo GPS, velocidade angular, aceleração linear e pressão são coletadas por meio da IMU, e o sonar é utilizado na obtenção de informações de altitude necessárias para o procedimento de pouso.

A interface com os atuadores é de responsabilidade do segundo subsistema. O VANT possui dois motores *brushless* acoplados as hélices, compondo assim os conjuntos propulsores. Eles são controlados por meio dos ESCs, que recebem como entrada a velocidade de referência de rotação. Os conjuntos estão acoplados aos servomotores, capazes de rotacionar todo o conjunto de propulsão. Estes dispositivos recebem como entrada o ângulo de inclinação desejado.

O terceiro subsistema é responsável pelo controle, onde são processados os algoritmos responsáveis pelo calculo do vetor de sinais de controle[1] do VANT com base nas informações recebidas dos sensores. Os algoritmos de controle, baseados na trajetória de referência, calculam os sinais adequados a serem aplicados aos atuadores (ângulos para os servomotores e velocidade angular para os motores *brushless*).

Os dois primeiros subsistemas são implementados na plataforma embarcada de baixo nível (STM-32-H407), e o terceiro subsistema é executado na plataforma embarcada de alto nível (Beaglebone).

Visando representar o funcionamento destes subsistemas, bem como projetar toda a estrutura do software, um diagrama de blocos foi desenvolvido, com o objetivo de realizar a representação funcional e a realização de simulações.

4.2.1 O modelo em diagrama de blocos

O modelo em diagrama de blocos foi desenvolvido na ferramenta Simulink, uma plataforma de modelagem e simulação de sistemas. Além da possibilidade de conexão com o ambiente MATLAB, este oferece aos desenvolvedores uma ampla biblioteca de blocos, onde diferentes sistemas podem ser projetados [MathWorks].

A modelagem do ambiente do VANT compreende dois subsistemas, o primeiro responsável pela representação das ações da estação base e o se-

[1] O vetor de sinais de controle descreve as forças e os torques a serem aplicados aos atuadores do VANT.

gundo pelas funcionalidades do VANT (Figura 15). Estes subsistemas são interligados por meio de um canal de comunicação wireless. Aliado a estes subsistemas se tem um terceiro componente responsável por representar os sinais recebidos via controle remoto.

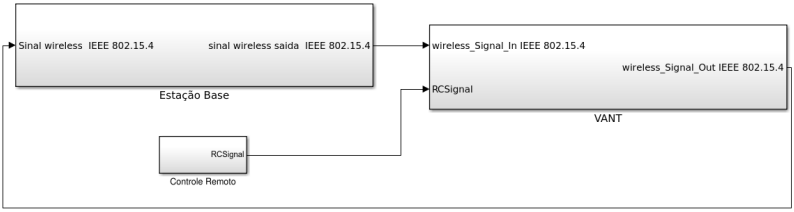

Figura 15 – Modelagem nível básico.

Os subsistemas tanto do VANT quanto da estação base são gerenciados por dois módulos de lógica discreta, responsáveis por inicializar e supervisionar o funcionamento geral dos sistemas. Esses são representados por duas máquinas de estado presentes em cada um dos subsistemas.

O subsistema da estação base apresentado na Figura 15, é detalhado na Figura 16 sendo composto por dois módulos, o de comunicação e o de lógica discreta. A arquitetura de comunicação é responsável pela codificação e decodificação das mensagens encaminhadas ao VANT, enquanto os eventos de lógica discreta são representados no segundo módulo.

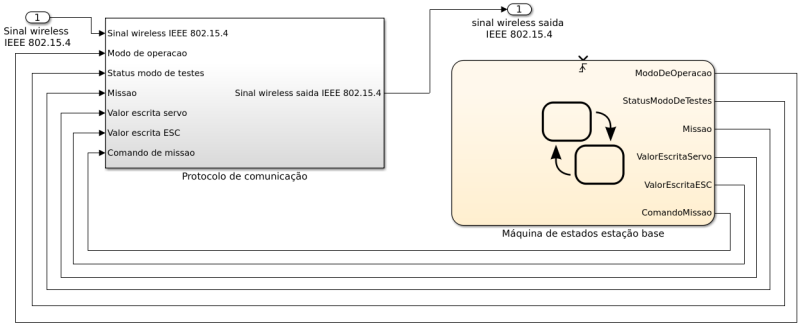

Figura 16 – Modelagem estação base.

Considerando a ocorrência dos eventos discretos na estação base, acionados pela interação do usuário, as configurações do VANT são realizadas.

Inicialmente o usuário pode definir o modo de operação do VANT, esse é enviado ao módulo de comunicação que encaminha ao VANT por meio do sinal *modo de operação*. Três opções são disponibilizadas: o modo de voo autônomo, o modo rádio controlado e o modo de testes.

Se o modo de voo autônomo é definido, uma missão deve ser configurada para execução. Nesta, a trajetória da aeronave é definida por meio de *waypoints*. Os dados referentes à missão são enviados ao VANT por meio da variável *missão* definida no protocolo de comunicação, após o correto recebimento e configuração da missão, esse encaminha uma mensagem de resposta sinalizando à estação base que está pronto para iniciar.

Neste momento tudo que o VANT necessita é de confirmação da estação base sinalizando o inicio da missão. Esta mensagem é encaminhada por meio da variável *comando de missão* definida no protocolo de comunicação, que permite ao usuário sinalizar tanto o inicio quanto o cancelamento da missão. A partir deste instante a aeronave ativa seu sistema de controle contínuo e a base passa apenas a monitorar o andamento da missão, verificando o estado dos sensores e enviando mensagens de controle ao VANT.

A partir da evolução da missão no modo de voo autônomo, o operador pode cancelar a sua execução a qualquer momento, solicitando assim o retorno da aeronave a sua posição de origem. Caso algum problema técnico venha a ocorrer, um módulo de segurança é ativado, abortando a execução da missão. Na ativação deste evento o VANT envia uma mensagem de alerta a estação base e realiza um pouso de emergência.

No modo de testes, o operador tem acesso aos dados, tanto puros (sem o fusionamento) quanto processados de todos os sensores e atuadores do sistema. Além disto comandos também podem ser enviados diretamente aos atuadores. A estação base sinaliza tanto o inicio quanto a finalização da atuação no modo de testes via a variável *Status modo de testes*, e atua enviando valores aos servomotores e aos ESCs por meio das variáveis *valor escrita servo* e *valor escrita ESC*. A operação do VANT neste modo se restringe ao envio das leituras a estação base, e atuação conforme os comandos recebidos. Neste modo a aeronave não realiza voos.

O detalhamento do subsistema do VANT apresentado inicialmente na Figura 15 é descrito na Figura 17. Este é composto pelos seguintes módulos: uma máquina de estados, que implementa o subsistema de lógica discreta; o módulo de comunicação *wireless*, responsável pela troca de mensagens com a estação base; o módulo de controle contínuo, responsável pelo controle de voo autônomo; o módulo de rádio controle, que processa os comandos recebidos via controle remoto; o módulo de processamento de dados, onde a comunicação com sensores e atuadores é realizada; o módulo de testes, responsável pela atuação direta da estação base sobre o VANT; o sistema de

monitoramento, responsável pela verificação da temperatura e nível de bateria visando garantir a integridade da aeronave; a rotina de testes, que é executada antes do inicio da missão visando verificar o funcionamento dos atuadores; o módulo de missão, responsável pelo controle da missão executada; o módulo de telemetria, o qual recebe as leituras da aeronave e as encaminha a estação base; e os demais módulos auxiliares, responsáveis pela representação dos sinais dos sensores e atuadores do sistema.

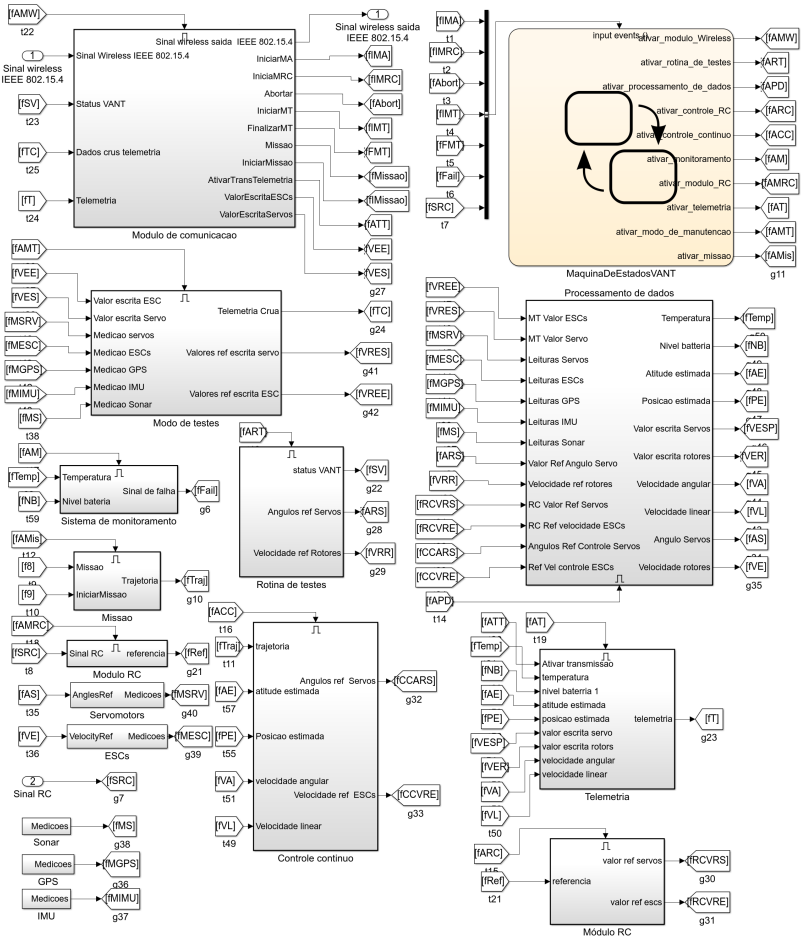

Figura 17 – Modelagem VANT.

A estrutura funcional do subsistema do VANT é gerenciada por meio

da máquina de estados, que é responsável pela ativação dos diferentes módulos do sistema de acordo com o modo de operação selecionado. A Figura 18 apresenta os modos principais, autônomo, testes e rádio controlado, assim como, os modos auxiliares como pouso de emergência e retornar a base.

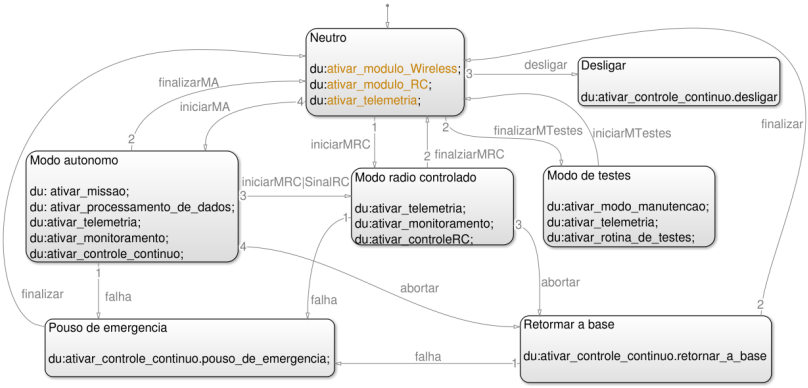

Figura 18 – Máquina de estados VANT.

Inicialmente o VANT se apresenta no estado *Neutro*, aguardando a seleção do modo de operação por parte da estação base. Neste estado os módulos *wireless*, de rádio controle e de telemetria são ativados.

O modo autônomo é definido pela ocorrência do evento *Modo autonomo*, na ativação deste estado são ativados os seguintes módulos do sistema: o módulo de missão; o módulo de processamento de dados; o módulo de monitoramento; e o módulo de controle contínuo.

A partir deste estado três eventos podem ocorrer: um evento de falha, que ativa o estado de *pouso de emergência* ativando também a função pouso de emergência do módulo de controle contínuo; um evento *abortar*, sinalizado pela estação base, ativando o evento *Retornar a base* e a função retornar a base no módulo de controle contínuo; e o estado *Modo rádio controle*, ativado tanto pela ocorrência de um sinal de rádio controle (*SinalRC*), ou pela inicialização via estação base por meio do evento *IniciarMRC*.

No estado *Modo rádio controlado* são ativados os módulos de *telemetria*, *monitoramento* e *controleRC*. Além dos estados descritos, a partir do estado neutro outros dois estados podem ser alcançados, o estado *Desligar* que sinaliza o desligamento físico da aeronave, e o estado *Modo de testes*, que descreve as ações do usuário da estação base diretamente sobre o VANT, onde os módulos de *manutenção*, *telemetria* e *rotina de testes* são ativados.

No modo rádio controlado, os sinais de referência são encaminhados

por meio do controle remoto, e então convertidos em referências de velocidade e inclinação enviadas aos atuadores. Desta forma, a aeronave é conduzida sem a necessidade do envio de uma missão. O módulo responsável por este controle é apresentado na figura 19.

Esse módulo recebe como entrada o sinal do controle remoto já decodificado, estes valores de referência são inseridos como entrada para o algoritmo de controle de estabilidade executado neste módulo. Como saída desta operação são gerados os valores a serem enviados aos servomotores e aos ESCs.

A ativação do modo rádio controlado pode ser realizada tanto por meio da sua seleção via estação base, quanto de forma automática, visto que durante a execução de uma missão, caso o VANT receba um sinal de rádio controle, este modo é ativado automaticamente.

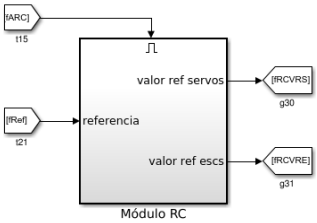

Figura 19 – Módulo rádio controlado.

O modo de operação autônomo é ativado após a definição da missão e sua operação ocorre sem interferências externas. A missão recebida contém uma trajetória pré-definida que é utilizada como referência pelo módulo de controle contínuo. Este subsistema é composto por algoritmos responsáveis por estimar a posição e orientação da aeronave baseados nos dados enviados pelos sensores. O detalhamento do subsistema de controle é apresentado na Seção 2.3.

A comunicação com os sensores e atuadores é estabelecida pelo subsistema de processamento. Nele ocorrem as transformações dos dados puros em informações processadas. Esse módulo também implementa o filtro de Kalman, que tem por objetivo fusionar as leituras da IMU com as do GPS visando aprimorar a estimação da posição da aeronave.

Estabelecer um canal confiável para troca de mensagens entre a estação base e o VANT é responsabilidade do módulo de comunicação. Para assegurar a comunicação é implementado o protocolo *wireless* padrão IEEE 802.15.4. As funcionalidades de codificação e decodificação das mensagens que trafegam neste meio, assim como a garantia de entrega segura, também são implementadas neste módulo.

O módulo de monitoramento é responsável por garantir a integridade física do VANT, sendo esta estabelecida pela verificação de alguns parâmetros do sistema, tais como, temperatura e nível de bateria. Caso algum destes se mostre fora da faixa de valores aceitáveis, um sinal de falha é emitido e a aeronave realiza um pouso de emergência no local seguro mais próximo a sua localização atual, evitando assim danos estruturais.

Na decorrência da execução de uma missão, a qualquer momento o operador da estação base pode solicitar a ativação da transmissão dos dados de telemetria (Figura 20), que é realizada pelo recebimento do sinal *ativar transmissão*. Assim, a estação base passa a receber as leituras dos sensores e atuadores do VANT, os dados referentes ao seu comportamento e da execução da missão.

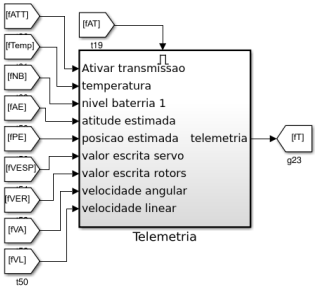

Figura 20 – Módulo de telemetria

Com base na metodologia aplicada a este projeto e a partir da especificação da modelagem funcional aliada à representação matemática do VANT, algumas análises de comportamento do sistema foram realizadas. No entanto para verificação e ajustes do comportamento dos algoritmos de controle, além de análises do comportamento da aeronave, a construção de um ambiente seguro de HIL que permita a realização destes processos se faz necessária. A seção a seguir descreve o processo de modelagem e desenvolvimento do HIL aplicado ao ambiente do ProVant.

4.3 PROJETO DO HIL PARA O PROVANT

Visando garantir que o processo de desenvolvimento dos sistemas embarcados transcorra de forma segura, a realização de testes é necessária para a verificação do atendimento aos requisitos de projeto. Assim, experimentos realizados em ambiente de simulação têm por objetivo analisar o comporta-

mento do sistema e viabilizar a sua aplicação na plataforma definitiva [Cai et al. 2008a].

No processo de simulação, modelos matemáticos são desenvolvidos para representar tanto o sistema de controle, quanto o modelo comportamental. Estes são concebidos com auxílio de ferramentas de simulação, como o Simulink. Variações destes ambientes são apresentadas de forma a possibilitar a integração da plataforma embarcada real com o modelo computacional. Neste ambiente, as leis de controle são aplicadas diretamente a plataforma embarcada que é conectada ao simulador descrevendo um modelo em HIL [Shixianjun et al. 2006].

O HIL permite a implementação de diferentes dinâmicas de simulação, visando verificar o comportamento do sistema em meio a diferentes estímulos, os quais não podem ser verificados de outra forma sem colocar em risco a integridade física dos projetos. Diferentes estratégias de controle podem ser executadas neste ambiente que provê um sistema capaz de interfacear com o modelo da planta real [Chandhrasekran et al. 2009].

Considerando o diagrama de blocos desenvolvido no Simulink descrito na Seção 4.2.1, um ambiente HIL foi desenvolvido com base na modelagem funcional, visando à simulação do VANT e a aplicação das técnicas de controle diretamente na plataforma embarcada. Este capítulo descreve o desenvolvimento dessa plataforma de emulação, onde diferentes testes podem ser propostos e características como utilização da plataforma, estabilidade do controlador, entre outras podem ser verificadas.

4.3.1 Ambiente HIL no escopo do VANT

A aplicação de algoritmos de controle na plataforma embarcada é um processo complexo, e exige uma cuidadosa verificação do comportamento e de ajustes. A necessidade de testes de forma a garantir altos níveis de segurança e confiabilidade exige um ambiente de emulação, onde estes processos sejam realizados com segurança.

O processo de transcrição dos algoritmos para a plataforma embarcada, assim como ajustes desses são facilitados por meio do uso do HIL, que permite a emulação do sistema no ambiente de testes. Nesse, além das verificações tradicionais realizadas com as análises via Simulink, se torna possível inserir perturbações tanto ao modelo quanto ao sistema de controle, e analisar questões relacionadas à plataforma embarcada como comportamento, capacidade computacional, tempos de resposta entre outras.

No contexto do VANT a utilização do HIL se mostra interessante, pois permite ao desenvolvedor a realização de análises sem colocar em risco a

integridade física da aeronave, bem como das pessoas envolvidas. Assim, são descritos nesta seção: a estrutura do ambiente de simulação; e a solução experimental proposta para verificação do limiar de segurança do sistema.

4.3.2 Estrutura do HIL

O ambiente experimental desenvolvido busca reproduzir as características do VANT real em ambiente de simulação. O sistema desenvolvido foi dividido em dois subsistemas: o primeiro é responsável pela execução dos algoritmos de controle implementado na plataforma embarcada (Beaglebone); o segundo tem por objetivo a representação do comportamento do VANT, sendo executado em um computador por meio do Simulink. Estes são interconectados por um link de comunicação de rede protocolo TCP/IP conforme apresentado na Figura 21.

O protocolo TCP/IP foi definido para utilização no HIL devido à abstração deste provida pelo Simulink, além da confiabilidade na entrega das mensagens garantida pelo protocolo.

Figura 21 – Estrutura *Hardware-in-the-loop*.

A sua concepção foi iniciada com a implementação dos algoritmos de controle do Simulink para a plataforma embarcada. Este processo não é trivial, pois o processo manual de transformação exige um esforço adicional dos desenvolvedores de forma a reproduzir fielmente na plataforma alvo o modelo implementado na ferramenta de simulação. A transformação dos modelos ocorreu com sua implementação na linguagem C++ com uso da biblioteca de *pthreads*.

Os algoritmos de controle foram transcritos conforme especificado na modelagem matemática (Seção 2.3) e com base no modelo inicialmente gerado no Simulink.

Aliado ao software embarcado, um modelo computacional foi desen-

volvido no Simulink de forma a representar o comportamento do VANT (Figura 22). Ele recebe os sinais de controle e de perturbação e têm como saída atitude, posição, orientação e as velocidades do VANT.

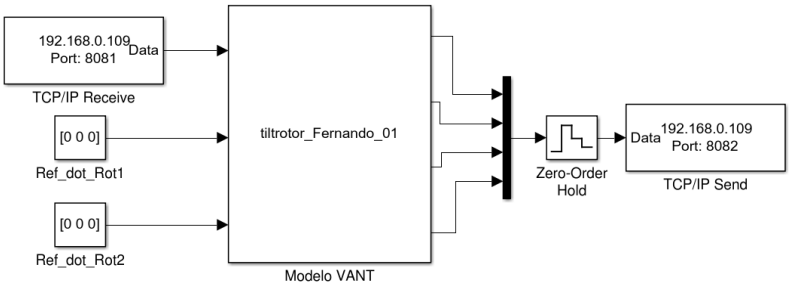

Figura 22 – Modelo VANT Simulink.

Este modelo recebe como entrada três vetores, os quais descrevem as entradas do modelo computacional. O primeiro desses é enviado pela plataforma embarcada, recebido pelo bloco *TCP/IP Receive*, e descreve às forças f_r e f_l e os ângulos dos servos α_r e α_l. Já os outros dois blocos de entrada (*Ref_dot_Rot1* e *Ref_dot_Rot2*), descrevem perturbações lineares e angulares ao sistema. Visto que perturbações não estão sendo aplicadas ao modelo, somente ao controle, estes vetores são enviados com valor zero.

A partir destas entradas são gerados quatro vetores de saída, os quais são concatenados e enviados à plataforma embarcada por meio do bloco *TCP/IP Send*. O primeiro destes vetores descreve a posição (X, Y e Z), o segundo apresenta a velocidade linear ($\dot{X}$, $\dot{Y}$ e $\dot{Z}$), o terceiro descreve os ângulos da aeronave (ϕ, θ e ψ) e o quarto apresenta as variações angulares ($\dot{\phi}$, $\dot{\theta}$, e $\dot{\psi}$).

O bloco responsável pela representação do modelo foi implementado por meio da biblioteca S-function do Simulink, que permite o desenvolvimento de funções diretamente nas linguagens de programação C ou C++. A partir da sua concepção, uma compilação é gerada no ambiente Matlab e o executável resultante é então incluído no modelo Simulink.

Nesta estrutura, cenários de avaliação foram propostos para verificação de diferentes configurações de forma a estabelecer um limiar seguro de execução dos algoritmos de controle, mantendo estabilidade e seguimento de trajetória na execução de voos autônomos.

4.3.3 Implementação do HIL e tarefas relacionadas

A implementação do HIL visa conceber o ambiente de simulação que agrega o modelo computacional da aeronave, desenvolvido no Simulink e o software aplicado à plataforma embarcada. Neste ambiente se busca aplicar os algoritmos de controle à plataforma embarcada, variar seu nível de utilização de CPU e analisar o comportamento tanto do modelo quanto do controle.

A estrutura de *software* aplicada à plataforma embarcada é composta por três tarefas: a primeira, de maior prioridade, executa os algoritmos de controle contínuo; a segunda estabelece a comunicação entre a plataforma embarcada e o modelo do VANT desenvolvido no Simulink; e a terceira, de menor prioridade, busca introduzir carga computacional a plataforma embarcada, onde a partir da variação de seu parâmetro de entrada, essa função apresenta diferentes níveis de utilização.

A tarefa de carga tem por objetivo introduzir carga computacional a CPU da plataforma embarcada, ou seja, por meio desta com a variação do seu parâmetro de entrada diferentes cenários de utilização podem ser emulados. O incremento de carga ocorre pela implementação do algoritmo da sequência Fibonacci, onde quanto maior o valor de entrada inserido, maior o tempo de execução, e por sua vez uma maior perturbação exercida sobre o sistema. Com aplicação dessa tarefa ao sistema, análises podem ser realizadas com intuito de estabelecer qual o limite máximo de utilização que a CPU da plataforma embarcada pode ser submetida, de forma que o comportamento do sistema de controle não seja afetado.

A comunicação da plataforma embarcada com a ferramenta de simulação ocorre via protocolo TCP/IP. Por meio do uso de *sockets* ela estabelece um servidor, que é conectado pelo Simulink em tempo de execução.

Para o gerenciamento das tarefas executadas, visto que o sistema operacional utilizado não é de tempo real, foi implementado uma tarefa principal. Semáforos são utilizados para a liberação das demais tarefas de acordo com o período das mesmas.

Com base nas análises do modelo de blocos e visando garantir a estabilidade e sincronia na execução das tarefas, os parâmetros de execução do sistema foram definidos, assim como, o tempo de execução e a utilização[2] de cada uma das tarefas foi verificado. Detalhes de especificação são apresentados na Tabela 1.

Neste ambiente se têm a tarefa de mais alta prioridade definida como

[2]O percentual de utilização das tarefas é obtido pela relação entre o tempo de execução dessa e seu período.

Controle que implementa a estratégia de controle, essa é definida com um período de 3 milissegundos e descreve uma utilização de CPU de 10,35%. A segunda tarefa descrita como *Comunicação* implementa o protocolo TCP/IP para troca de mensagens com o modelo Simulink, essa é definida com um período de 12 milissegundos e descreve uma utilização de CPU de 70,29 %. A terceira tarefa descrita como *Carga Computacional* busca incrementar o nível de utilização de CPU, sendo definida com um período de 12 milissegundos, e sua utilização de CPU variando entre 0,06% à >100% de acordo com o nível de carga aplicado.

Tabela 1 – Conjunto de tarefas.

Tarefa	**Prio.**	**Período**	**T. Execução (ms)**	**Utilização (%)**
Controle	1	3 ms	0,3104	10,35
Comunicação	2	12 ms	8,4342	70,29
Carga Computacional	3	12 ms	0,0076 (min) 39,5719 (max)	0,06 (min) > 100 (max)

Estabelecido o ambiente HIL, diferentes configurações tanto de controle quanto de modelo puderam ser inseridas e seu comportamento analisado. Visando estabelecer um limiar seguro para execução dos algoritmos de controle, diferentes níveis de utilização de CPU foram propostos no ambiente da plataforma embarcada.

As diferentes configurações aplicadas à plataforma embarcada foram estabelecidas com a definição de uma faixa de valores atribuídos à tarefa de carga computacional. Desta forma a cada valor aplicado a essa tarefa se têm um cenário de emulação com uma carga computacional diferente. Apesar dessa possuir uma prioridade mais baixa que a de controle, a influência exercida sobre o ambiente afeta o desempenho de todas as tarefas do sistema.

Os valores atribuídos à tarefa de carga computacional variam entre 0 e 500000, o que corresponde a diferentes níveis de utilização de CPU produzidos. O desempenho da tarefa de carga de acordo com esta variação é apresentado no Capítulo 5.

Estabelecido o ambiente de simulação HIL, e realizadas análises visando estabelecer o limiar de segurança utilização de CPU para a execução dos algoritmos de controle, abordagens alternativas foram propostas com a aplicação dos AA de forma a verificar o comportamento do sistema. Os detalhes dessa nova abordagem são descrito na sessão a seguir.

4.4 PROJETO DAS TAREFAS UTILIZANDO *ANYTIME ALGORITHMS*

A definição do limiar de segurança para execução de um sistema embarcado crítico é comumente realizada com base nos piores casos de execução deste (do inglês *Worst-Case Execution Time* - WCET). No entanto, a especificação de sistemas baseados neste critério pode resultar na subutilização da arquitetura de hardware, pois a execução do WCET nem sempre ocorre.

Devido a este fato, a adição de novas tarefas a esta arquitetura muitas vezes é inviabilizada, frente à restrição de recursos da arquitetura de hardware. Assim como, o suporte a um conjunto de tarefas especificado pode ser prejudicado de forma a ocasionar a perda de *deadlines*.

Buscando estabelecer um ambiente capaz de gerar uma melhor utilização da CPU, diferentes técnicas de computação podem ser empregadas. Isto ocorre por meio de abordagens diferenciadas das tradicionais baseadas no WCET. Dentre estas temos as técnicas de computação imprecisa, mais especificamente os AA.

Com base na metodologia utilizada, aliada ao ambiente HIL desenvolvido, duas abordagens da utilização dos AA são propostas a seguir, visando verificar a viabilidade da sua aplicação no ambiente do VANT. As propostas descrevem aplicação dos AA nas tarefas de controle, assim como da sua utilização nas tarefas de mais baixa prioridade do sistema. As características dos ambientes desenvolvidos são descritas a seguir.

4.4.1 Ambiente experimental

A aplicação dos AAs no contexto do VANT tem por objetivo verificar o comportamento do sistema e analisar a viabilidade do seu emprego, seja no desenvolvimento de algoritmos de controle, ou na utilização em tarefas de menor prioridade. Desta forma, dois ambientes são propostos: o primeiro consiste na aplicação dos AA na tarefa de controle; já a segunda abordagem visa analisar o comportamento das tarefas de mais baixa prioridade, implementadas com base nos AAs.

A abordagem *Anytime* aplicada neste contexto se difere um pouco das descritas na literatura, devido ao fato da estratégia de controle adotada exigir uma baixa carga de processamento, assim como, da sua impossibilidade de divisão em um número maior de malhas de controle devido às limitações do controlador.

Desta forma, o controle *Anytime* aplicado consiste na divisão da tarefa de controle inicialmente desenvolvida para as simulações de estabelecimento do limiar de segurança, em duas tarefas: a primeira responsável pelo controle

de estabilidade definida com a prioridade mais alta do sistema; a segunda tem por objetivo a realização do controle de seguimento de trajetória, sendo definida com prioridade menor que a tarefa anterior.

Neste contexto o uso dos AA ao invés de apresentar uma implementação com múltiplas malhas, descreve um ambiente onde a manutenção da estabilidade do VANT é sempre garantida, enquanto o seguimento de trajetória é executado somente quando há tempo de CPU disponível devido a sua menor prioridade.

Inicialmente, a divisão da tarefa de controle foi implementada e analisada no Simulink, verificando assim a viabilidade da sua divisão e da definição dos períodos para cada uma das tarefas. A partir desta validação as tarefas foram codificadas e transferidas para a plataforma embarcada.

Aliado à estratégia de controle, foram adicionadas as tarefas responsáveis tanto pela comunicação quanto pela adição de carga de CPU ao ambiente. A estrutura aplicada foi à mesma utilizada no ambiente de simulação e estabelecimento do limiar de segurança, e o conjunto de valores de carga também se manteve o mesmo. As características do conjunto de tarefas desenvolvido são descritos na Tabela 2.

Tabela 2 – Conjunto de tarefas Controle Anytime.

Tarefa	**Prio.**	**Período**	**T. Execução (ms)**	**Utilização (%)**
C. Estabilidade	1	3 ms	0,3287	10,96
C. Trajetória	2	12 ms	0,0420	0,35
Comunicação	3	12 ms	8,4342	70,29
Carga Computacional	4	12 ms	0,0076 (min) 39,5719 (max)	0,06 (min) $>$100 (max)

Nesta temos a tarefa de mais alta prioridade definida como *Controle de Estabilidade* definida com um período de 3 milissegundos, a qual descreve uma utilização de CPU de 10,96%. A segunda tarefa deste conjunto implementa o *Controle de Trajetória*, sendo definida com um período maior que a anterior (12 milissegundos) e descreve uma utilização de 0,35% da CPU. A definição de períodos distintos para as malhas de controle que anteriormente eram executas em uma mesma tarefa nos garante uma maior flexibilidade ao sistema.

As tarefas de *Comunicação* e *Carga Computacional* foram implementadas neste conjunto de tarefas com as mesmas características do cenário anterior, onde se mantém os mesmos períodos e os níveis de utilização de CPU

aplicados ao sistema.

A segunda abordagem proposta descreve a utilização do mesmo ambiente aplicado às simulações de estabelecimento do limiar de segurança, no entanto com a inclusão dos AA neste contexto. Esta ocorre por meio da implementação de uma tarefa adicional de mais baixa prioridade, que descreve a execução do método Newton-Raphson.

O método Newton-Raphson foi inserido com objetivo de verificar o comportamento *anytime* aplicado às tarefas de mais baixa prioridade, visando assim à liberação de CPU para as tarefas mais prioritárias. Este método é empregado na localização das aproximações das raízes de uma função $f(x)$. Com base na aproximação inicial da raiz x_i, torna-se possível estender uma linha tangente para f no ponto $(x_i, f(x_i))$. O ponto onde esta linha tangente intercepta o eixo x geralmente representa uma estimativa melhorada da raiz [Chapra e Canale 2008].

O método Newton-Raphson é empregado na definição da raiz da seguinte função $f(x)$:

$$f(x) = (x * x) + log(cos(x)^2). \tag{4.1}$$

Para obtenção da raiz de (4.1), o método requer o cálculo da sua derivada, que é descrita pela equação:

$$f'(x) = 2 * (x - tan(x)). \tag{4.2}$$

O método Newton-Raphson é baseado nos critérios de parada ε_1 e ε_2 e seus valores podem variar de acordo com o tempo de CPU disponível. A tabela 3 descreve os níveis de precisão sobre os quais o método pode ser aplicado. De acordo com o número de iterações executadas, maior o nível de precisão do resultado apresentado pelo método, assim como, o tempo de processamento necessário aumenta.

Tabela 3 – Níveis de precisão Newton-Raphson.

Grau	**Precisão**	**Iterações**
1	0.1	9
2	0.001	20
3	0.00001	24
4	0.0000001	28
5	0.000000001	32
6	0.00000000001	36

A abordagem *anytime* implementada neste método se baseia em seus critérios de parada (ε_1 e ε_2). Sua iteração é iniciada com a aplicação do menor grau de precisão e este é incrementado de acordo com a disponibilidade do processador. Na aplicação deste ao ambiente do HIL foi definido que os critérios de parada receberiam os mesmos valores, ou seja, $\varepsilon_1 = \varepsilon_2$. Portanto, as execuções da tarefa foram realizadas utilizando sempre o mesmo valor de entrada, com x_0 definido como $46,5238$.

O conjunto de tarefas aplicado ao segundo ambiente, assim como, as características descritas são apresentados na tabela 4.

Tabela 4 – Conjunto de tarefas *Anytime*.

Tarefa	**Prio.**	**Período**	**T. Execução (ms)**	**Utilização (%)**
Controle	1	3 ms	0,3104	10,35
Comunicação	2	12 ms	8,4342	70,29
Carga Computacional	4	12 ms	0,0076 (min) 39,5719 (max)	0,06 (min) >100 (max)
Anytime	5	3 ms	0,3291 (min) 1,0199 (max)	10,97 (min) 34,00 (max)

Neste conjunto de tarefas especificado as tarefas de *Controle*, *Comunicação* e *Carga Computacional* foram implementadas com as mesmas características do ambiente de definição do limiar de segurança, ou seja, os períodos tempos de execução e percentuais de utilização de CPU permaneceram inalterados. No entanto foi realizada a inclusão de uma quinta tarefa (*Anytime*) que implementa o método Newton-Raphson, essa foi definida com um período de 3 milissegundos e apresenta uma utilização de CPU variando de 10,97% a 34%.

As simulações foram realizadas com base nos três conjuntos de tarefas especificados, aplicando a variação dos níveis de carga de CPU e buscando analisar o comportamento do sistema. Os resultados e as análises realizadas com base nestes são apresentadas no próximo capítulo.

5 RESULTADOS E AVALIAÇÃO

A partir da concepção do ambiente de emulação em HIL, cenários de avaliação foram desenvolvidos com objetivo de verificar o comportamento do modelo computacional da aeronave, do sistema de controle contínuo e da plataforma embarcada.

Conforme apresentado no Capítulo 4, três conjuntos de tarefas foram especificados: o primeiro, descrito na Tabela 1, têm por objetivo analisar o comportamento do sistema de controle e estabelecer um limite de utilização de CPU segura para execução do sistema; o segundo, apresentado na Tabela 2, descreve uma proposta de controle baseada nos AA, sua simulação têm por objetivo verificar a viabilidade de aplicação dessa técnica às tarefas mais críticas do sistema; e o terceiro, especificado na Tabela 4, apresenta uma proposta *anytime* aplicado, à tarefa de menor prioridade do sistema, desta forma se têm por objetivo analisar a viabilidade de aplicação dessa técnica aplicada às tarefas de menor prioridade do sistema.

Os detalhes de simulação de cada um destes cenários, os resultados obtidos e a análise do comportamento destes ambientes são descritos a seguir.

5.1 SIMULAÇÕES AMBIENTE DE PROJETO SEGURO

Com base no conjunto de tarefas especificado na Tabela 1 uma faixa de valores foi especificada para a tarefa computacional de forma a produzir diferentes níveis de utilização de CPU atribuídos à plataforma embarcada. Neste cenário se têm por objetivo analisar o comportamento do sistema de controle e estabelecer um limite de utilização de CPU segura para execução do sistema;

Para cada faixa de valor definido foram realizadas dez execuções, e a partir do processamento dos dados gerados, a análise de alguns fatores como, a variação do número de execuções, o tempo de execução, e o desvio padrão, foi realizada.

Os valores aplicados à tarefa de carga computacional variam na faixa entre 0 a 500000, com incremento de 25000 para cada valor de simulação.

O comportamento da tarefa de carga computacional é apresentado na Figura 23. Nessa se pode verificar que inicialmente não há variação na quantidade de execuções desta tarefa, porém a partir da carga 100000 uma oscilação maior na quantidade de execuções desta tarefa é verificada.

O desvio padrão verificado nesta tarefa se mantém constante, variando entre 200000 e 250000 e a partir de 400000. Nestes pontos uma variação

maior na quantidade de execuções desta tarefa também é verificado. O tempo de execução desta tarefa é incrementado de acordo com a elevação do nível de carga aplicado a tarefa.

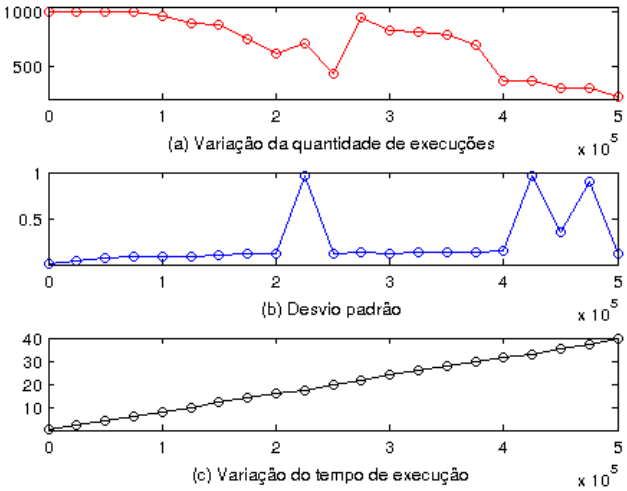

Figura 23 – Desempenho da tarefa de carga computacional.

O desempenho da tarefa de controle também foi analisado com base nos valores de referência da tarefa carga computacional, verificando, assim, questões como estabilidade e segurança com o aumento de utilização da CPU. Os resultados obtidos são apresentados na Figura 24.

Analisando o comportamento da tarefa de controle, se pode verificar inicialmente que essa se mantém estável, visto que não é verificada variação na quantidade de execuções da tarefa em todos os níveis de carga de CPU aplicados. O tempo de execução também sem mantém sem grandes variações, quando analisado seu comportamento e o desvio padrão.

O processo de simulação em HIL tem como premissa básica a obrigatoriedade no estabelecimento da comunicação entre os dois subsistemas provida por meio do padrão TCP/IP. No entanto, esse canal de comunicação está sujeito a incertezas, sejam estas do meio onde está inserido, ou da própria abstração descrita pela ferramenta de simulação que não nos permite ajustes em seu protocolo desenvolvido.

Os resultados apresentados descrevem que no limiar inicial de utilização de CPU estabelecido (entre 0 e 100000), o sistema de controle do VANT

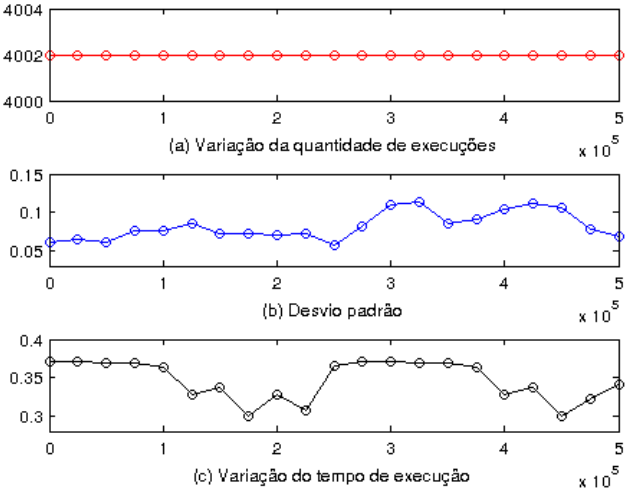

Figura 24 – Desempenho da tarefa de controle.

se manteve estável, descrevendo a trajetória da aeronave, a mantendo estável e sem a perda de execuções da tarefa de controle. No entanto, a partir do aumento da utilização da CPU (entre os valores 100000 e 500000), as tarefas com mais baixa prioridade tiveram seu desempenho prejudicado. Assim, quanto maior a utilização inserida no ambiente, menor o número de execuções das tarefas de menor prioridade.

A variação apresentada no tempo de execução da tarefa de controle, ocorre pelo fato desta fazer uso de variáveis compartilhadas com a tarefa de comunicação, onde o bloqueio exercido pelo *mutex* e das incertezas apresentadas pela tarefa de comunicação, considerando que não se tem domínio do protocolo implementado na ferramenta de simulação, a tarefa de controle descreve pequenas variações no tempo de execução da tarefa.

Outro fator a ser considerado tem relação ao tempo de execução das tarefas, onde foi verificado que estes consideram as interferências apresentadas pelo sistema, como bloqueios e atrasos de ativação entre outros. Desta forma, as análises com relação ao tempo de computação são baseadas no tempo total de execução da tarefa e não somente no tempo real de uso de CPU.

Com base nos resultados apresentados, foi verificado que o mecanismo desenvolvido para controle e ativação das tarefas não se mostrou muito eficaz, pois apesar de controlar a ativação, ele não realiza a interrupção de seu

processamento. Desta forma, conforme verificado na tarefa de carga computacional, em alguns casos o tempo de execução descreve valores superiores ao período descrito para a mesma.

Considerando as incertezas apresentadas pelo ambiente de rede, a variação no tempo de execução devido aos bloqueios e a ineficácia apresentada pelo mecanismo de escalonamento, pode-se verificar que, conforme apresentado na Figura 24, não há perda na quantidade de execuções da tarefa de controle. Desta forma o sistema mantém a aeronave estável mesmo em situações de sobrecarga.

A partir da realização das simulações foi possível verificar que dentro do limite de utilização de CPU estabelecido, o desempenho da tarefa de controle, mais crítica do sistema, se mantém estável, assim como a aeronave mantém a trajetória definida e estabilizada em seu ponto de destino. No entanto, apesar da aeronave se manter em voo, o desempenho das tarefas de menor prioridade do sistema são prejudicadas.

Visando reduzir a perda de desempenho apresentado nas tarefas de mais baixa prioridade, uma abordagem alternativa utilizando *Anytime Algorithms* foi proposta com objetivo de melhorar o desempenho mesmo em ambientes com tempo reduzido de utilização de CPU. Os detalhes referentes à sua aplicação são descritos na próxima seção.

5.2 SIMULAÇÕES *ANYTIME*

O segundo ambiente de simulação foi configurado com base no conjunto de tarefas especificado na Tabela 2, a faixa de valores especificada para a tarefa computacional aplicada ao primeiro cenário se manteve a mesma nestas simulações, visando produzir diferentes níveis de utilização de CPU atribuídos à plataforma embarcada.

Para cada faixa de valor definido dez execuções foram realizadas. Os valores aplicados à tarefa de carga computacional variam na faixa entre 0 a 500000, com incremento de 25000 para cada simulação.

A simulação deste cenário têm por objetivo verificar a viabilidade de aplicação dos AA às tarefas mais críticas do sistema, e o comportamento do controlador adotado.

O comportamento da tarefa de carga computacional neste segundo se descreveu com as mesmas características apresentadas na Figura 23, visto que não houveram modificações tanto na sua implementação quanto nos parâmetros de entrada aplicados.

Neste cenário o controle da aeronave é dividido em duas tarefas, o controle de estabilidade e o controle de seguimento de trajetória. Inicialmente,

buscou-se analisar o comportamento da tarefa referente ao controle de estabilidade, sendo esta essencial para a realização dos voos. A Figura 25 descreve seu comportamento de acordo com os níveis de carga de CPU aplicados.

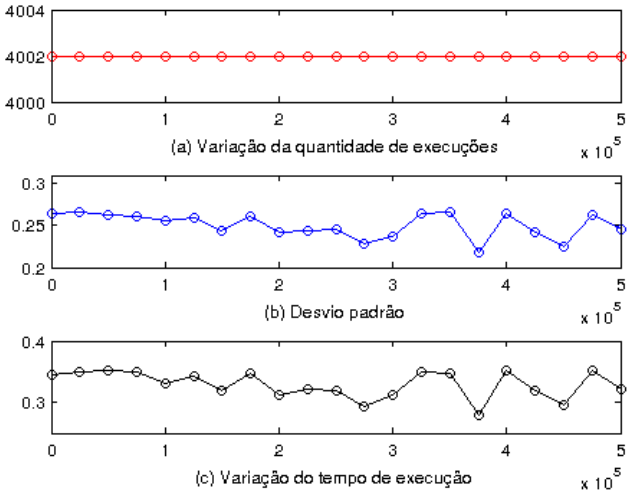

Figura 25 – Desempenho da tarefa de controle de estabilidade.

A execução do controle de estabilidade se mantém constante durante a aplicação de todos os níveis de carga de CPU aplicados, desta forma é verificado que não há variação na quantidade de execuções desta tarefa.

O tempo de execução do controle de estabilidade também não apresenta variações significativas dentre os diferentes níveis de carga de CPU aplicados, e considerando seu desvio padrão é possível verificar um tempo de execução com variações muito pequenas.

O desempenho apresentado pelo controle de seguimento de trajetória também foi verificado. A Figura 26 apresenta o comportamento desta com base nos níveis de carga aplicados.

Analisando o comportamento da tarefa de controle de seguimento de trajetória, é possível verificar que apesar desta possuir uma prioridade menor que a tarefa anterior, esta também se manteve estável durante todo o processo de simulação. A quantidade de execuções desta se manteve a mesma durante todos os níveis de carga aplicados.

O tempo de execução da tarefa de controle de seguimento de trajetória se mostrou muito curto, se comparado ao tempo de execução da tarefa de

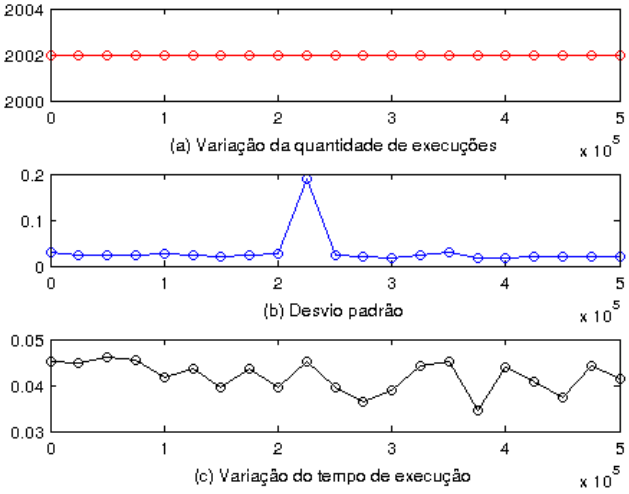

Figura 26 – Desempenho da tarefa de controle de seguimento de trajetória.

controle de estabilidade, porém, apesar deste se mostrar um pouco maior em alguns pontos, seu tempo de processamento se manteve sempre na faixa entre 0,035 ms a 0,048 ms. Considerando o desvio padrão, esta amplitude de valores não descreve uma variação considerável do tempo de processamento.

Apesar da divisão da estratégia de controle em duas tarefas apresentada como proposta *anytime*, analisando os resultados apresentados, se pode verificar que o tempo de execução das tarefas de controle implementadas neste cenário são muito próximos aos obtido no ambiente onde a implementação foi realizada em somente uma tarefa. Desta forma, não houve um ganho significativo na sua implementação. No entanto, um ganho de flexibilidade na aplicação deste controlador é verificado, visto que se têm duas malhas trabalhando com períodos distintos.

Deve-se considerar que a carga de processamento utilizada pela tarefa do controle de trajetória é muito pequena se comparada ao restante do sistema. Desta forma, seria necessário verificar o comportamento dos AA aplicados a uma estratégia de controle onde o seguimento de trajetória apresenta uma carga maior de processamento.

O terceiro ambiente de simulação descreve o comportamento dos AA aplicados à tarefa de mais baixa prioridade do sistema. Neste contexto, foi utilizado o ambiente descrito nas simulações de estabelecimento do limiar de

segurança com a adição da tarefa AA.

Com base no conjunto de tarefas especificado na Tabela 4 se têm por objetivo analisar o comportamento da tarefa *anytime*.

Neste cenário os valores aplicados à tarefa de carga computacional variam na faixa entre 0 a 300000, com incremento de 25000 para cada valor de simulação. Para cada faixa de valor definido foram realizadas dez execuções.

Desta forma, o comportamento da tarefa *anytime*, que implementa o método Newton-Raphson, pode ser analisado. A Figura 27 descreve o desempenho desta de acordo com os níveis de carga aplicados.

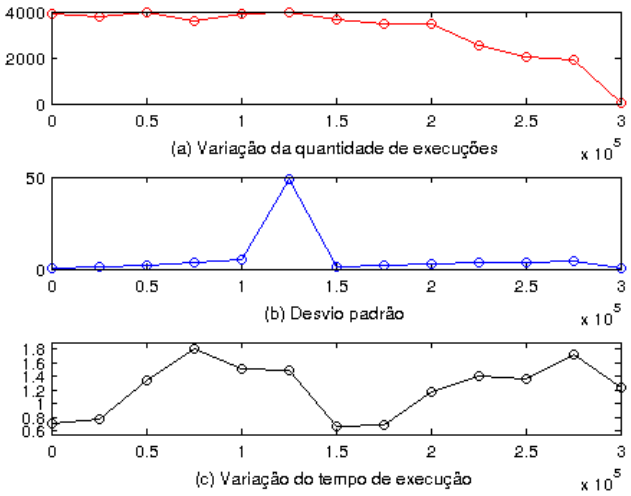

Figura 27 – Desempenho da tarefa *anytime*.

Nesta tarefa é possível verificar que até a aplicação do nível de carga de 200000, não há uma grande variação na quantidade de execuções desempenhada por esta tarefa, no entanto a partir deste valor a quantidade de execuções da tarefa é reduzida consideravelmente, chegando bem próximo à zero.

O tempo de execução descrito pela tarefa apresenta uma grande variação de acordo com o nível de carga de CPU aplicado, este fato ocorre tanto pela aplicação dos diferentes níveis de precisão implementados na tarefa, quanto pelo fato desta apresentar a menor prioridade do sistema, o que acaba afetando seu desempenho. O desvio padrão se mantém constante variando apenas entre 100000 e 150000, o que reforça a variação dos tempos de execução

exercidas pelos fatores já mencionados.

A quantidade de execuções e os níveis de precisão aplicados a cada uma dessas são apresentados nas Figuras 28 e 29.

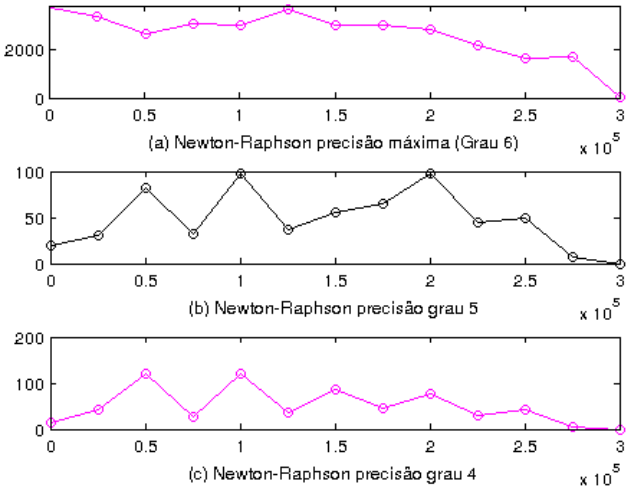

Figura 28 – Níveis de precisão aplicados à tarefa *anytime*.

Inicialmente na Figura 28(a) são apresentadas as execuções onde o nível máximo de precisão é aplicado à tarefa *anytime*, neste gráfico se pode verificar que há uma variação na quantidade de execuções em todos os níveis de carga aplicados. Conforme o nível de utilização de CPU é incrementado também se verifica uma redução na quantidade de execuções com nível máximo de precisão.

Os níveis de precisão 5 e 4 apresentados nas Figuras 28(b) e (c) descrevem quantidades de execuções semelhantes, apresentando variação em todos os níveis de utilização de CPU, assim como a quantidade de execuções utilizando cada um destes níveis descreve resultados semelhantes.

Já a Figura 29 descreve os níveis mais baixos de precisão aplicados à tarefa *anytime* e os resultados onde o método não apresentou uma saída válida.

Os níveis de precisão 3, 2 e 1 apresentados nas Figuras 29(a), 29(b) e 29(c) descrevem níveis de execuções semelhantes, variando em todos os níveis de carga de CPU aplicados e apresentando quantidades bem reduzidas da sua aplicação a partir do nível de carga 200000.

A Figura Figuras 29(d) apresenta as execuções onde o método não executou durante tempo suficiente para que um resultado, mesmo que com precisão mínima fosse produzido.

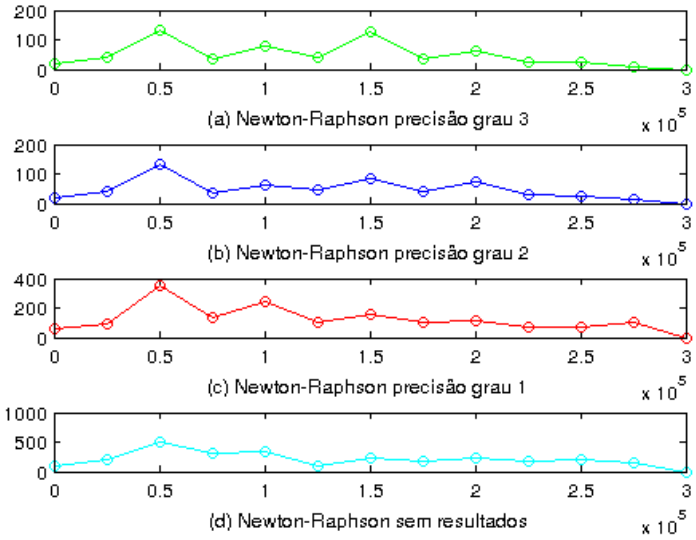

Figura 29 – Níveis de precisão aplicados à tarefa *anytime*.

Por meio da simulação se pode verificar um comportamento onde, de acordo com o tempo de processamento disponível, diferentes níveis de precisão foram aplicados. A variação referente ao tempo de CPU pode ser verificada analisando os tempos de execução de acordo com o nível de carga de CPU aplicado, considerando que em todos os níveis aos quais este cenário foi submetido variações na quantidade de execuções da tarefa foram verificadas, assim como a aplicação de níveis intermediários de precisão.

Apesar de algumas execuções do método Newton-Raphson aplicado à tarefa *anytime* não apresentarem resultados (representadas pelas execuções com zero iterações), e de algumas tarefas perderem seus deadlines (descrito pela variação no total de execuções em cada nível de carga aplicado), os resultados apresentados descrevem um bom desempenho da tarefa *anytime*.

Se considerado a implementação tradicional do algoritmo, onde somente o nível máximo de precisão é aplicado (grau 6), o desempenho desta tarefa seria muito prejudicado, visto que haveria um número maior de de-

adlines perdidos, pois os resultados com precisão intermediária não seriam considerados. Esta redução na quantidade de execuções da tarefa poderia ocasionar uma perda de desempenho do sistema, levando em consideração a contribuição que essa exerce no conjunto de tarefas.

Considerando a aplicação dos AA no contexto do VANT, esta técnica se apresenta como uma alternativa viável, visando uma melhor utilização do processador. No entanto, testes com algoritmos que exijam uma carga computacional maior, e alguns ajustes tanto no mecanismo de escalonamento, quanto nos algoritmos aplicados a plataforma embarcada, devem ser realizados, buscando sua implantação no ambiente real do VANT.

Além do comportamento de diferentes conjuntos de tarefas aplicadas a plataforma embarcada, o comportamento da aeronave, representado pelo modelo computacional, também foi avaliado, sendo apresentado a seguir.

5.3 COMPORTAMENTO DA AERONAVE

Com base nas entradas de força e ângulos dos servos, o modelo computacional implementado no Simulink descreve o comportamento linear e angular da aeronave de acordo com os sinais aplicados.

Nas simulações realizadas em todos os cenários, para verificar o comportamento do sistema de controle de seguimento de trajetória, a aeronave foi especificada em sua posição inicial como em zero (para todos os eixos X, Y e Z), e atribuído como referências ao controlador os sinais de 0.1 para o eixo X, -0.1 para o eixo Y e 0.1 para o eixo Z. A atribuição de valores tão próximos se deve ao tempo de resposta esperado da aeronave que nos obrigou a ajustar o controlador para receber pontos bem próximos uns dos outros.

O comportamento do sistema de controle de estabilidade é verificado com a atribuição do valor zero as referências dos ângulos ϕ, θ e ψ, os quais representam a aeronave estável.

Os três cenários de simulação executados descreveram o mesmo comportamento da aeronave. A representação do comportamento linear da aeronave é apresentado na Figura 30, onde a partir da posição inicial, essa atinge os valores de referência estabelecido para os três eixos um segundo após o inicio da atuação do sistema de controle.

O comportamento angular da aeronave é descrito na Figura 31, onde é possível observar que o ângulo de guinada apresenta uma dinâmica de estabilização mais rápida, enquanto os ângulos de rolagem e de arfagem necessitam de um tempo em torno de 1 segundo para estabilizar.

Com base nas simulações realizadas e analisando os resultados apresentados nos três cenários propostos, se pode verificar que os algoritmos de

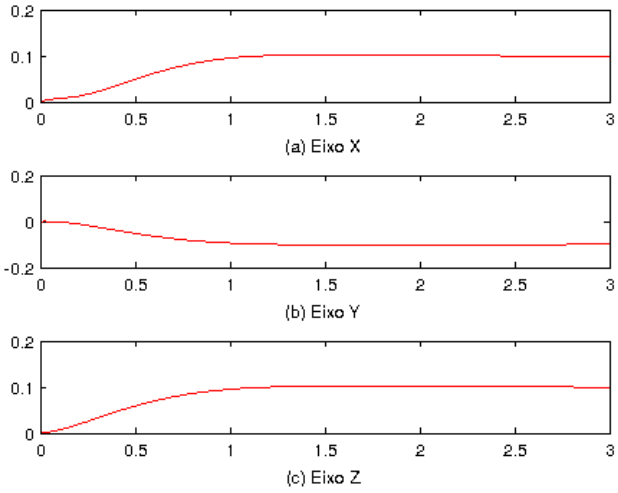

Figura 30 – Comportamento linear da aeronave.

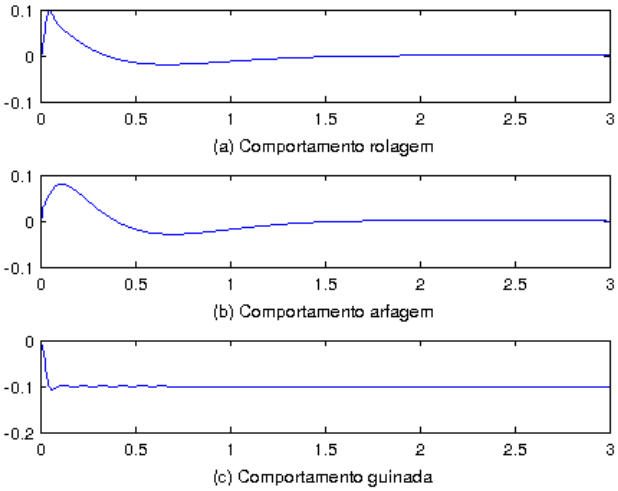

Figura 31 – Comportamento angular da aeronave.

controle apresentaram desempenho satisfatório em todas as simulações. Da mesma forma que o modelo computacional desenvolvido representou com fidelidade o comportamento da aeronave real.

6 CONCLUSÃO E TRABALHOS FUTUROS

Esta dissertação apresentou o projeto e desenvolvimento da arquitetura de software do sistema embarcado do VANT, assim como um ambiente de emulação que agrega segurança ao processo de planejamento de sistemas embarcados críticos. O ambiente de emulação, descreve uma abordagem bem próxima do ambiente real visando à realização de testes e ajustes nos sistemas implementados antes da sua aplicação no ambiente definitivo.

Durante a realização desta pesquisa se pôde compreender as técnicas de projeto de software voltadas à modelagem de *Cyber-Physical-Systems*, assim como as etapas de processo necessárias para sua modelagem.

A arquitetura do sistema computacional embarcado foi modelada, e por meio desta as demais funcionalidades envolvidas também puderam ser projetadas e avaliadas. Desta forma, obteve-se uma arquitetura robusta e confiável a ser aplicada ao ambiente do VANT.

A complexidade inerente do desenvolvimento destes sistemas foi compreendida, assim como, a necessidade do desenvolvimento de mecanismos capazes de permitir o projeto seguro de aplicações voltadas ao controle embarcado de VANTs.

Com o desenvolvimento do ambiente de emulação HIL, foi possível descrever e avaliar o comportamento tanto dos sistemas de controle quanto da aeronave em modo de voo. Desta forma o estabelecimento de um limiar de segurança de utilização de CPU para execução da estratégia de controle adotada foi estabelecido, sem que a integridade física da aeronave fosse colocada em risco.

A partir da concepção deste ambiente abordagens alternativas de projeto foram propostas com aplicação da técnica dos *Anytime Algorithms*. Assim, estas puderam ser testadas e avaliadas de forma a permitir um melhoramento tanto no contexto da estratégia de controle adotada, quanto das tarefas auxiliares do sistema. Apesar dos resultados obtidos não descreverem ganhos significativos quanto ao desempenho do sistema embarcado, visto que os tempos de execução das tarefas de controle em ambos os ambientes foram muito próximos, a solução *anytime* se mostrou mais flexível, permitindo a adição de novas tarefas, e mantendo o mesmo nível de estabilidade avaliado no primeiro ambiente.

Esta pesquisa resultou em uma arquitetura de software proposta ao sistema embarcado de um VANT, de forma a garantir o cumprimento dos requisitos de tempo real impostos pelo sistema de controle, e um ambiente de emulação capaz de permitir a aplicação de diferentes abordagens de controle e de diferentes cenários nos quais o VANT possa ser submetido.

Neste sentido, diversas são as alternativas para desenvolvimento de trabalhos futuros envolvendo o ambiente do VANT como um todo, tanto no contexto da arquitetura de software, como no aperfeiçoamento do ambiente de simulação, dentre as alternativas destacam-se:

(a) O desenvolvimento de novas estratégias de controle aplicadas ao ambiente de simulação desenvolvido;

(b) O aperfeiçoamento do mecanismo de escalonamento desenvolvido para realizar a ativação das tarefas na plataforma embarcada, visando um maior controle de ativação e interrupção das tarefas agendadas;

(c) A análise da viabilidade de substituição do modelo computacional de representação do VANT desenvolvido no Simulink, por um modelo desenvolvido em linguagem computacional C++, de forma a extrair um melhor desempenho nas simulações com altos níveis de carga aplicada, visto que, conforme verificado com níveis de carga reduzidos o sistema se apresenta de maneira estável;

(d) Aperfeiçoar o meio de comunicação entre os ambientes de emulação do HIL, de forma a reduzir as incertezas apresentadas;

(e) Integrar o ambiente de simulação com um software de simulação de voo, de forma que seja possível analisar graficamente o comportamento da aeronave durante todas as etapas de voo.

REFERÊNCIAS

BEAGLEBOARD. **Beaglebone Website**. 2013. http://beagleboard.org/Products/BeagleBone.

BECKER, L. B.; FARINES, J.; BODEVEIX, J.; FILALI, M.; VERNADAT, F. Development process for critical embedded systems. In: **Workshop de Sistemas Embarcados, Gramado. Anais. Porto Alegre: SBC**. [S.l.: s.n.], 2010. p. 95–108.

BERTHOMIEU, B.; BODEVEIX, J.-P.; FARAIL, P.; FILALI, M.; GARAVEL, H.; GAUFILLET, P.; LANG, F.; VERNADAT, F. Fiacre: an Intermediate Language for Model Verification in the Topcased Environment. In: **ERTS 2008**. Toulouse, France: [s.n.], 2008. <http://hal.inria.fr/inria-00262442>.

BUDIYONO, A. Advances in unmanned aerial vehicles technologies. In: **International Symposium on Intelligent Unmanned System**. [S.l.: s.n.], 2008. p. 1–13.

CAI, G.; CHEN, B.; LEE, T.; DONG, M. Design and implementation of a hardware-in-the-loop simulation system for small-scale uav helicopters. In: **Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on**. [S.l.: s.n.], 2008a. p. 29–34.

CAI, G.; FENG, L.; CHEN, B. M.; LEE, T. H. Systematic design methodology and construction of {UAV} helicopters. **Mechatronics**, v. 18, n. 10, p. 545 – 558, 2008b. ISSN 0957-4158.

CHAKI, S.; CLARKE, E.; OUAKNINE, J.; SHARYGINA, N.; SINHA, N. State/event-based software model checking. In: BOITEN, E.; DERRICK, J.; SMITH, G. (Ed.). **Integrated Formal Methods**. [S.l.]: Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 2999). p. 128–147. ISBN 978-3-540-21377-2.

CHANDHRASEKRAN, V. K.; PRASAD, R. B.; CHOI, E.; MIN, D. Hardware-in-the-loop simulation of uav non-linear control system of mini-helicopter. In: **ICHIT'09**. [S.l.: s.n.], 2009. p. 288–295.

CHAPRA, S.; CANALE, R. **Métodos Numéricos para Engenharia**. [S.l.]: McGraw Hill Brasil, 2008.

CHOWDHURY, A. B.; KULHARE, A.; RAINA, G. Back-stepping control strategy for stabilization of a tilt-rotor uav. In: **Control and Decision Conference (CCDC), 2012 24th Chinese**. [S.l.: s.n.], 2012. p. 3475–3480.

DEAN, T.; BODDY, M. An analysis of time-dependent planning. In: **Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)**. Saint Paul, Minnesota, USA: AAAI Press/MIT Press, 1988. p. 49–54. ISBN 0-262-51055-3.

DONADEL, R.; RAFFO, G.; BECKER, L. Modeling and control of a tiltrotor uav for path tracking. In: **19th World Congress of the International Federation of Automatic Control (IFAC 2014)**. [S.l.: s.n.], 2014.

FEILER, P. H.; GLUCH, D. P.; HUDAK, J. J. **The Architecture Analysis & Design Language (AADL): An Introduction**. [S.l.], 2006. <http://www.aadl.info/aadl/currentsite/currentusers/notation.html>.

GONCALVES, F.; BODANESE, J.; DONADEL, R.; RAFFO, G.; NORMEY-RICO, J.; BECKER, L. Small scale uav with birotor configuration. In: **Unmanned Aircraft Systems (ICUAS), 2013 International Conference on**. [S.l.: s.n.], 2013a. p. 761–768.

GONCALVES, F.; DONADEL, R.; RAFFO, G.; BECKER, L. Assessing the use of simulink on the development process of an unmanned aerial vehicle. In: **3rd Workshop on Cyber-Physical Systems (CyPhy 2013)**. [S.l.: s.n.], 2013b.

GONCALVES, F.; RAFFO, G.; BECKER, L. Analyzing the use of anytime algorithms on an unmanned aerial vehicle. In: **3o. Simpósio Brasileiro de Eng. de Sistemas Computacionais (SBESC 2013)**. [S.l.: s.n.], 2013c.

HENZINGER, T.; MANNA, Z.; PNUELI, A. **Timed Transition Systems**. 1992.

HOFFMANN, G. M.; HUANG, H.; WASL, S. L.; TOMLIN, E. C. J. Quadrotor helicopter flight dynamics and control: Theory and experiment. In: **In Proc. of the AIAA Guidance, Navigation, and Control Conference**. [S.l.: s.n.], 2007.

IEEE. **IEEE Standard 802.15.4**. 2011. `http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf`.

JENSEN, J.; CHANG, D.; LEE, E. A model-based design methodology for cyber-physical systems. In: **Wireless Communications and Mobile**

Computing Conference (IWCMC), 2011 7th International. [S.l.: s.n.], 2011. p. 1666 –1671.

JUNG, D.; TSIOTRAS, P. Modeling and hardware-in-the-loop simulation for a small unmanned aerial vehicle. In: **Aerial Vehicle," AIAA Infotech at Aerospace, Rohnert Park, CA, May 2007, AIAA**. [S.l.: s.n.], 2007. p. 07–2763.

KIM, M.; STEHR, M.-O.; TALCOTT, C. A distributed logic for networked cyber-physical systems. In: **ELSEVIER Journal of Science of Computer Programming (SCP)**. [S.l.]: Springer-Verlag, 2012. (FSEN'11), p. 190–205. ISBN 978-3-642-29319-1.

LEE, E. Cyber physical systems: Design challenges. In: **Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on**. [S.l.: s.n.], 2008. p. 363 –369.

LIU, C. L.; LAYLAND, J. W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. **J. ACM**, ACM, New York, NY, USA, v. 20, n. 1, p. 46–61, jan. 1973. ISSN 0004-5411.

LIU, J.; SHIH, W.-K.; LIN, K.-J.; BETTATI, R.; CHUNG, J.-Y. Imprecise computations. **Proceedings of the IEEE**, v. 82, n. 1, p. 83 –94, jan 1994. ISSN 0018-9219.

LOUALL, R.; BELLOULA, A.; DJOUADI, M.; BOUAZIZ, S. Real-time characterization of microsoft flight simulator 2004 for integration into hardware in the loop architecture. In: **Control Automation (MED), 2011 19th Mediterranean Conference on**. [S.l.: s.n.], 2011. p. 1241–1246.

MANGHARAM, R.; SABA, A. Anytime algorithms for gpu architectures. In: **Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd**. [S.l.: s.n.], 2011. ISSN 1052-8725.

MATHWORKS. **Mathworks MATLAB Simulink**. `http://www.mathworks.com/products/simulink/`.

MAXBOTIX. **Especificações Maxbotix MB1200**. 2013. `http://www.maxbotix.com/Ultrasonic_Sensors/MB1200.htm`.

MICROCHIP. **Especificações Mrf24j40mc**. 2013. `http://ww1.microchip.com/downloads/en/DeviceDoc/75002A.pdf`.

MIKROKOPTER. **BL-Ctrl V2.0 datasheet**. 2013. `http://www.mikrokopter.de/ucwiki/en/BL-Ctrl_2.0?highlight=%28bl-ctrl%29`.

MODELMOTORS. **Especificações motor AXI 2814/20**. 2013. http://www.modelmotors.cz/index.php?page=61&product=2814&serie=20&line=GOLD.

NARAYANAN, V.; PHILLIPS, M.; LIKHACHEV, M. Anytime safe interval path planning for dynamic environments. In: **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on**. [S.l.: s.n.], 2012. p. 4708–4715. ISSN 2153-0858.

OLIMEX. **Especificações STM32-H407**. 2013. https://www.olimex.com/Products/ARM/ST/STM32-H407/.

OLIVARES-MENDEZ, M.; MONDRAGON, I.; CAMPOY, P.; MARTINEZ, C. Fuzzy controller for uav-landing task using 3d-position visual estimation. In: **Fuzzy Systems (FUZZ), 2010 IEEE International Conference on**. [S.l.: s.n.], 2010. p. 1–8. ISSN 1098-7584.

OPENSIMPAD, O. e OpenZaurus e. **Angstrom Linux**. 2013. http://www.angstrom-distribution.org/.

PINHEIRO, B. C.; MARTINS, D. L.; RICO, J. E. N.; BECKER, L. B. Desenvolvimento de sistemas de controle tempo real embarcados: Uma exploração acerca da metodologia e ferramentas. In: **XVIII Congresso Brasileiro de Automática (CBA2010)**. [S.l.: s.n.], 2010.

QUAGLI, A.; FONTANELLI, D.; GRECO, L.; PALOPOLI, L.; BICCHI, A. Designing real-time embedded controllers using the anytime computing paradigm. In: **Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on**. [S.l.: s.n.], 2009. p. 1–8. ISSN 1946-0759.

RAFFO, G. V.; ORTEGA, M. G.; RUBIO, F. R. Nonlinear $\mathscr{H}_\infty$ Controller for the Quad-Rotor Helicopter with Input Coupling. In: **18th World Congress, IFAC 2011**. Milano, Italy: [s.n.], 2011. p. 13834–13839.

ROBOTIS. **Especificações Dynamixel RX-24F**. 2013. http://support.robotis.com/en/product/dynamixel/rx_series/rx-24f.htm.

SCHMIDT, D. Guest editor's introduction: Model-driven engineering. **Computer**, v. 39, n. 2, p. 25–31, Feb 2006.

SHIXIANJUN; JIAKUN, S.; HONGXING, L. Hardware-in-the-loop simulation framework design for a uav embedded control system. In: **Control Conference, 2006. CCC 2006. Chinese**. [S.l.: s.n.], 2006. p. 1890–1894.

STEVAL. **Especificações Steval MKI124V1**. 2013. http://www.st.com/web/en/catalog/tools/PF253482.

UBLOX. **Especificações GPS LEA-6h**. 2013. http://www.u-blox.com/en/gps-modules/pvt-modules/lea-6-family.html.