DESENVOLVIMENTO DE AMBIENTE OPERACIONAL ROS PARA QUAD-ROTORES

Raphael de Abreu Alves e Silva
Mestrado/IFSP
Tec. Laboratório Elétrica/IFSP

Alexandre Simião Caporali
Doutorado/ USP
Professor/IFSP

RESUMO

O estudo de robôs tem se tornou recorrente no ambiente acadêmico, em setores profissionais e amadores, isto é alavancado pela redução dos custos de componentes e da disponibilidade de ferramentas gratuitas, tais como: ROS; Arduino; e Linux. Um dos grandes obstáculos dessa tarefa de elaborar sistemas robóticos é a integração dos diversos componentes construtivos, estes fabricados por diferentes empresas e com metodologias de uso diversos. Este trabalho visa a elaboração de um sistema operacional utilizando o ambiente de desenvolvimento ROS (*Robot Operating System* – Sistema Operacional para Robôs), o qual propõe a melhora da integração de subsistemas e componentes pertencentes aos robôs. Este ambiente favorece a concepção em diversas frentes concomitantes, mantendo a coesão do sistema. Outro fator fundamental é que ROS é um ambiente gratuito com grande comunidade colaborativa, oferecendo diversos módulos de funcionalidades específicas (operação remota, controle de posição) e exemplos disponíveis para o uso. A maior contribuição desse trabalho é a elaboração de um sistema que possa auxiliar na expansão de tecnologias aplicadas para robôs do tipo quad-rotor, possibilitando avanços mais rápidos e em harmonia com o que tem sido utilizado por outros pesquisadores.

Palavras-chave: Desenvolvimento de sistemas robóticos. Quad-rotor. ROS. Simulador de quad-rotores. Telemetria.

Introdução

Ultimamente, observa-se um crescimento do interesse em robótica nas mais diversas áreas industriais (automobilística, manufatura e espacial). Tem-se utilizado de robôs para substituir homens em locais perigosos, trabalho repetitivo e situações onerosas. Pesquisas dessa área são em equipamentos de plataforma aérea (aviões com asas fixas, dirigíveis, helicópteros, quad-rotores). Cada modelo tem suas vantagens e desvantagens, tornando necessária uma avaliação desses benefícios para a aplicação (BRESCIANI, 2008).

Equipamentos robóticos dependem da utilização de sensores para garantir sua operação, conseguindo assim executar tarefas de maneira precisa e segura, reduzindo riscos e retrabalhos. Devido ao avanço nos estudos dos MEMS (*Micro Electro Mechanical Systems*) Sistemas Micro eletromecânicos, os equipamentos têm seu tamanho

ISSN: 2526-4982

reduzido sistematicamente. Logo, a utilização desses itens tornou-se mais usual e houve aumento na produção, reduzindo custo final (MELO, SALLES e ALMEIDA, 2010).

Devido às mudanças de cenário houve um favorecimento às pesquisas com VANT's (Veículos Aéreos Não Tripulados) de tamanho reduzido, e observou-se crescimento significativo desse segmento, pois foi obtida melhoria na eficiência e precisão. Equipamentos alimentados por bateria, que possuem menor complexidade mecânica e não transportam combustível inflamável, tem sido mais estudados (MELO, SALLES e ALMEIDA, 2010).

Os aeromodelos, também denominados como VANT's, podem operar em situações perigosas, repetitivas, em condições hostis ou de difícil acesso ou onerosas. Observa-se, também, uma grande utilização em atividades civis e entretenimento (BOUABDALLAH; MURRIERI e SIEGWART, 2004);

Ao embarcar periféricos que possibilitam aquisição de imagens, podem ser utilizados como ferramentas no uso de exploração de ambientes, segurança e perseguições (Draganflyer, 2013), redução de custo em filmagens ou fotos panorâmicas (*Perspectives Aerials*, 2013), inspeções de linhas elétricas de difícil acesso, monitoração de animais, plantações e florestas (MELO, SALLES e ALMEIDA, 2010).

Como utilizar componentes diversos que apresentem falta de compatibilidade e integrar subsistemas genéricos concebidos independentemente do sistema no desenvolvimento de robôs do tipo quad-rotor?

Segundo o site da organização responsável pelo ROS (*Robot Operating System* – Sistema Operacional para Robôs), esse sistema é um ambiente flexível para desenvolvimento de códigos computacionais para robôs, uma série de ferramentas, bibliotecas e convenções visando simplificar a tarefa de desenvolver robôs complexos.

Devida à ampla comunidade que foi criada com o objetivo de ser colaborativa, utilizar ROS se torna atraente para realização dessa atividade. Há o benefício de diversas soluções estarem disponíveis, geradas por usuários especialistas de diversos lugares do mundo (DEMARCO; WEST e COLLINS, 2011).

Dessa forma, é possível o desenvolvimento desarticulado de um mesmo robô, tornando o sistema um integrador, possibilitando a adaptação das funcionalidades de um robô sem interferir umas nas outras, assim, mais de uma pessoa pode trabalhar no mesmo projeto, manter a integração e garantir que as atividades não se prejudiquem no resultado final (COUSINS *et al.*, 2010).

Segundo COUSINS *et al.* (2010), compartilhar códigos é uma prática comum atualmente. Busca-se com isso aumento da velocidade de desenvolvimento e permite-se replicar e melhorar os resultados de projetos disponibilizados. Esse é um fator decisivo para o uso de ROS na concepção de quad-rotores. Utilizando-se desse ambiente, pode-se concentrar em inovações pontuais e resolução de problemas específicos ou melhorias.

O objetivo geral deste trabalho é desenvolver um sistema operacional para um VANT genérico, possibilitando a implantação de novas funcionalidades e integração simplificada utilizando o *framework* ROS, linguagem de programação C++ e Linux.

Os objetivos específicos deste trabalho são:

- Implementar comunicação entre o quad-rotor que utiliza myRIO e o computador;

- Implementar comunicação entre o quad-rotor, Arduino e o computador;
- Programar o sistema computacional em ROS que interprete dados recebidos e possa enviar comandos ao quad-rotor;
- Implementar telemetria da plataforma real, possibilitando a validação de experimentos com VANT's.

MEYER *et al.* (2012) propõe que utilizando a ferramentas do ROS um ambiente pode-se desenvolver um ambiente de simulação de quad-rotores. Dessa forma, pode-se obter maior eficiência na concepção de tecnologias que usem VANT's desse tipo. O Gazebo/ROS foi utilizado porque oferece um ambiente confiável para simulação, simula baseado em diversas interações físicas do objeto simulado e possibilita ao usuário modificar parâmetros, um exemplo, as configurações do controlador.

Foi utilizado um modelo 3D desenhado no programa Blender (MEYER *et al.,* 2012), para a simulação geométrica e modelamento matemático para simulação cinemática e dinâmica. Para simulação dos sensores, foram utilizados códigos adicionais externos ao Gazebo, que podem ser ativados ou desativados, conforme a necessidade. Foram realizados experimentos que consistiam de trajetórias e transições de velocidades, tanto no modelo simulado, como no quad-rotor real, onde foi verificado um resultado satisfatório, que repetia o sistema real no ambiente computacional.

GRABE *et al.* (2013) apresentam um sistema para controlar multiplos VANT's simultaneamente e com comunicação bilateral na operação homem-máquina e máquina-máquina. O TeleKyb (nome dado ao projeto) possui diversos códigos que realizam operações específicas, por exemplo:

a) *Human Interface*: ambiente de operação do VANT por uma pessoa;
b) TeleKyb Base: oferece suporte para desenvolvedores de robôs;
c) TeleKyb Core: oferece uma biblioteca de controladores, estimadores e outras ferramentas;
d) ROS-Simulink Bridge: Oferece uma ferramenta para conexão do sistema com MATLAB.

Este projeto realizou experimentos e obteve bons resultados apresentando confiabilidade e integrabilidade com projetos diversos.

DUNKLEY *et al.* (2014) propuseram um nanoquad-rotor equipado com uma câmera e transmissão sem fio, comunicando com um computador no solo, o quad-rotor mais leve capaz de gerar imagens com baixo custo, robustez e fácil reconfiguração.

VANT's com peso e dimensões reduzidas têm se mostrado úteis para experimentos em locais onde há pouco espaço e risco de colisão. O projeto é disponibilizado integralmente para reprodução e também oferecidos kits para montagem e utilização.

Utilizou-se ROS no trabalho por ser uma plataforma que favorece o compartilhamento dos resultados e é um ambiente de fácil evolução. A plataforma tem baixo custo, 7 minutos de autonomia de voo e 20 minutos de tempo de recarga e mostrou-se muito resistente a quedas e choques. Através do ROS, é possível fazer telemetria dos estados do quad-rotor, com atraso de 8 ms, ferramenta de visão computacional confiável.

Foram feitos testes de voo com e sem câmera, na plataforma com bons resultados em ambos.

Algumas ferramentas integráveis ao ROS, também importantes no trabalho, serão utilizadas. Gazebo é um ambiente que possibilita simular dinâmica e cinemática dos robôs e comparar resultados com dados reais. Foi utilizado o pacote ROS Hector_quadrotor como ponto de início da concepção do sistema operacional para os quad-rotores genéricos. Esse projeto apresenta funcionalidades similares aos objetivos que este trabalho almejou e que compactua com o *framework* do ROS de reutilização de código e, também, os objetivos deste trabalho visam complementar esse pacote adicionando novas funções: Telemetria e integração com VANT's reais.

O trabalho progrediu em três etapas: adaptação do pacote Hector_quadrotor às características do projeto; simulação virtual e integração com modelo real. Foi necessário um computador com sistema operacional LINUX para utilizar ROS, Gazebo, rviz e Arduino e outro computador com ambiente Windows para LabVIEW. Um sistema de comunicação sem fio e um protótipo de quad-rotor utilizando myRIO ou Arduino.

Os códigos foram escritos em linguagem de programação C++/ROS. O sistema e seus subsistemas serão programados de maneira independente e foram integrados posteriormente ao ambiente principal. Essa estratégia permitiu o progresso descentralizado, no qual mais de um pesquisador pode trabalhar no projeto ao mesmo tempo. Foram utilizados programas de licença aberta, gratuita ou já adquiridos anteriormente e disponíveis ao grupo de pesquisa, não gerando nenhum novo custo ao projeto. Desejou-se, conforme apresentado na introdução, mostrar que a evolução descentralizada utilizando ROS otimiza o projeto e integração de subsistemas de quad-rotores.

Novamente, a concepção de plataformas do tipo quad-rotor tem se mostrado um tema recorrente e utilizar um ambiente que auxilia na evolução de robôs se mostra um catalisador deste processo. Em diversos artigos, é exaltado o uso de ROS, e um ambiente de simulação para o desenvolvimento de quad-rotores (MEYER *et al.*, 2012; DUNKLEY *et al.*, 2014; ALEJO *et al.*, 2014).

Pode ser observado a seguir um esquema de como o sistema foi disposto. O material utilizado foi:

a) Dois computadores: Um para executar o sistema operacional Linux e sistema ROS e outro com sistema operacional Windows para utilizar o LabVIEW na elaboração da integração entre o sistema ROS e o myRIO utilizado no protótipo;

b) LabVIEW: Ferramenta para integração do myRIO ao ROS;

c) ROS *Indigo Igloo*;

d) Gazebo 2;

e) Rviz;

f) Pacotes ROS: Hector_quadrotor, "ROS for LabVIEW Software", Rosserial e teleop_twist_keyboard;

g) Roteador: Utilizado para conexão entre o myRIO ou Arduino e o computador com Linux ou Windows;

h) Quad-rotor que utiliza myRIO ou Arduino;

Hector_quadrotor

Foi utilizado o pacote hector_quadrotor como base para o desenvolvimento do sistema utilizado nesse trabalho. Foram identificados os tópicos pelos quais quad-rotor simulado pelo pacote hector_quadrotor se comunicava:

/cmd_vel: Tópico que tem tipo geometry_msgs/Twister. Através dele pôde-se comunicar dados referentes a velocidade angular e linear do quad-rotor. Um exemplo da mensagem pode ser observado a seguir:

$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 1.0,y: 1.0, z: 3.0}, angular: {x: 0.1,y: 0.1,z: 0.1}}'

Apesar de existir o tópico /command/twist que seria a opção de comando mais óbvia com configuração idêntica ao /cmd_vel o sistema não respondeu como esperado, sendo necessário o uso do segundo no lugar do primeiro.

Incialmente, foram desenvolvidos códigos que publicassem nesses tópicos e pudesse ser observada a resposta no sistema. O sensor inercial do quad-rotor deveria ser lido e alimentar o sistema com esses dados colhidos no formato do tópico, de maneira a fazer uma estimativa de posição e estado do VANT e representação no ambiente de simulação. Foram utilizados algoritmos para converter o sinal de orientação, velocidade angular, aceleração linear e altura do sensor para o formato desejado.

Então foi desenvolvido na plataforma myRIO um código que gerou um publicador nos tópicos anteriormente descritos /cmd_vel, /command/pose e com os dados da IMU utilizados pela plataforma no programa LabVIEW. O código foi feito seguindo tutoriais da comunidade (myRIO Publisher, 2015).

Foi desenvolvido na plataforma myRIO um código que gere um leitor do tópico anteriormente descritos /cmd_vel/ de maneira que é possível fazer a operação do quad-rotor pelo computador com sistema ROS no programa LabVIEW, com o código seguindo tutoriais da comunidade (myRIO Subscriber, 2015).

Foi elaborado um código similar ao realizado para os testes utilizando o computador com LabVIEW para embarcar no myRIO e comunicar ao sistema ROS, porém não foi obtido nenhum sucesso e foram observadas as seguintes dificuldades:

- Não foi possível seguir os tutoriais, pois eles não funcionavam como o apresentado;
- Aparentemente, a falta de suporte à ferramenta impossibilitou a busca por soluções para os problemas com os tutoriais apresentados;
- Problemas de comunicação entre o myRIO e o computador com LabVIEW através da WI-FI, que não foi possível ser solucionado, forçando a comunicação somente através do cabo USB;

Com a impossibilidade de comunicação entre o myRIO e ROS, não foi possível, durante esse trabalho, integrar um quad-rotor que utilizava a plataforma myRIO ao sistema ROS.

Foram desenvolvidos códigos similares para a plataforma Arduino. Foi utilizada a IDE em Linux para programação reaproveitando porções de códigos de terceiros e orientações sobre utilização de drivers específicos para Arduino e ROS. Os códigos funcionaram no simulador e na placa duemilanove. Na placa Due, entretanto, não foi possível transferir o programa devido a incompatibilidades da IDE e ROS.

Devido às dificuldades encontradas para a comunicação entre as plataformas microcontroladoras e o sistema ROS, não foi possível realizar um teste que obtivesse os dados dos sensores e os enviasse ao sistema ROS, após o tratamento do sinal com filtro apropriado, de forma a visualizar a plataforma real em ambiente virtual de maneira satisfatória.

Testes em ambiente virtual

O sistema que foi desenvolvido durante esse trabalho pode ser acessado utilizando a sequência de linhas de comando abaixo, que inicializam e dão acesso às funcionalidades desenvolvidas; as Figuras 1, 2 e 3 mostram o sistema sendo iniciado no computador.

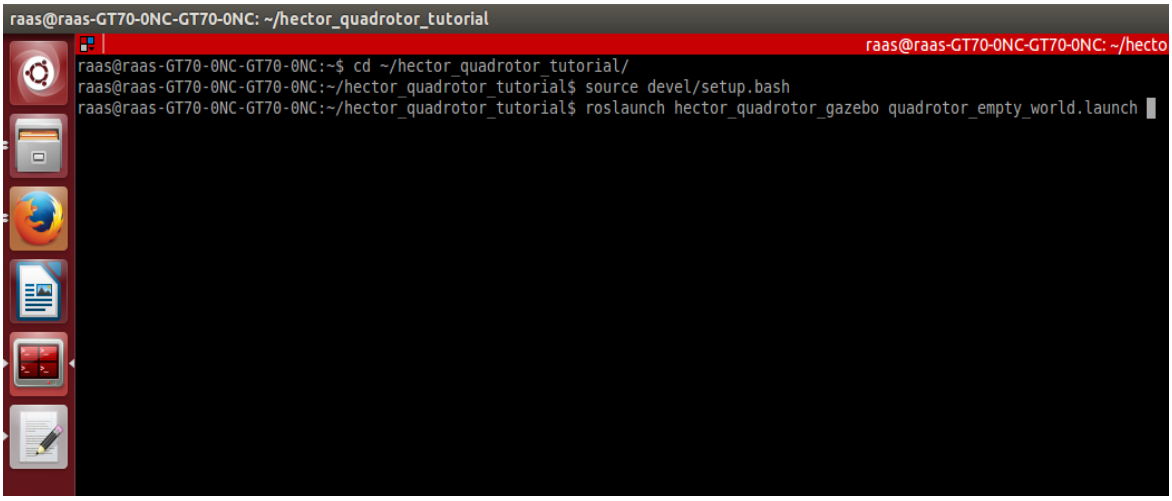

Figura 1: Tela de inicialização do sistema no terminal.

$ cd ~/hector_quadrotor_tutorial/

$ source devel/setup.bash

$ roslaunch hector_quadrotor_gazebo quadrotor_empty_world.launch

Foi alterado o arquivo de inicialização, adicionando funcionalidades diferentes do padrão (teleoperação, ambiente de simulação simplificado e reprodutor de trajetórias pré-definidas).

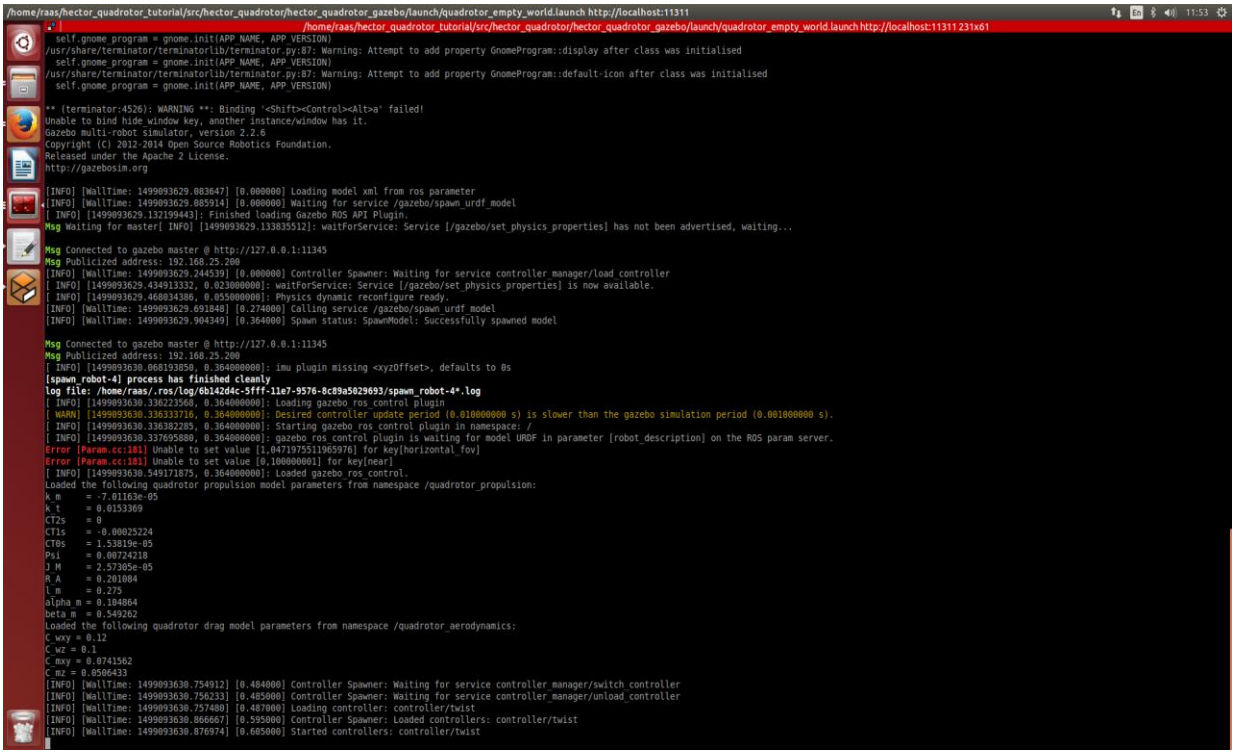

Figura 2: Tela após inicialização do sistema no terminal.

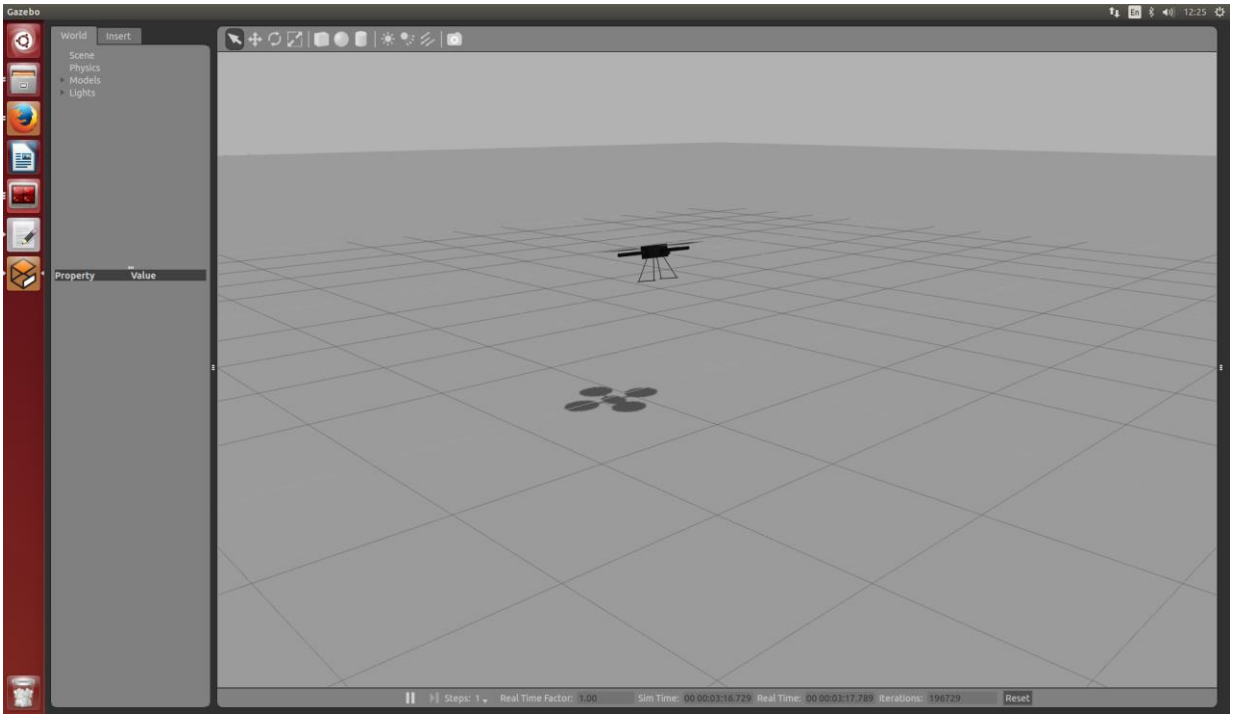

Figura 3: Tela do simulador de voo do quad-rotor.

Foram utilizados pacotes auxiliares para realizar a teleoperação do quad-rotor em ambiente virtual. Foi utilizado o pacote teleop_twist_keyboard, que permitiu a operação do quad-rotor pelo teclado do computador e se mostrou uma ferramenta simples e com bons resultados para observação e identificação dos parâmetros desejados do sistema, que pode ser observado na Figura 4.

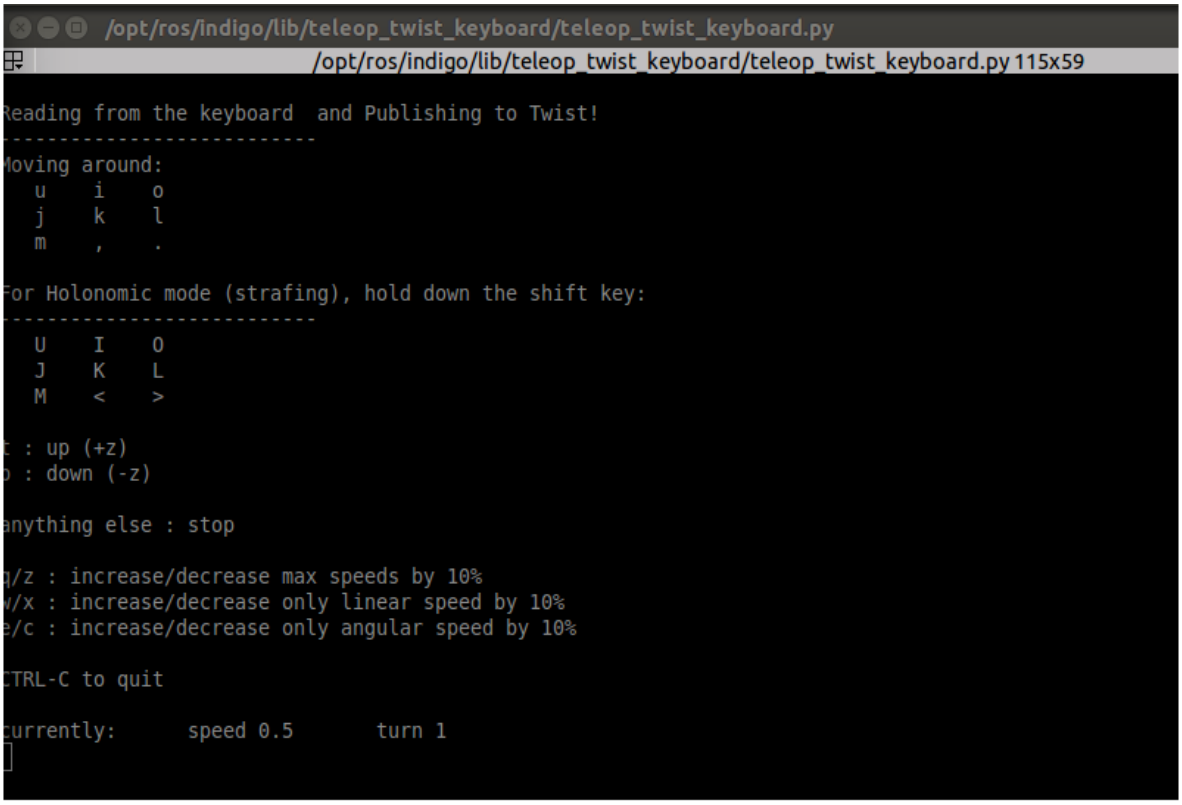

Figura 4: Tela de inicialização do comando de tele-operação por teclado do quad-rotor

Outro pacote, mais complexo que o anterior, utilizado para teleoperação, foi hector_quadrotor_teleop, que permite o uso de um controle de Xbox para operação do sistema. Utilizar um controle desse tipo é mais confortável e possibilita uma maior precisão e melhor operação do sistema, que pode ser observado na Figura 5.

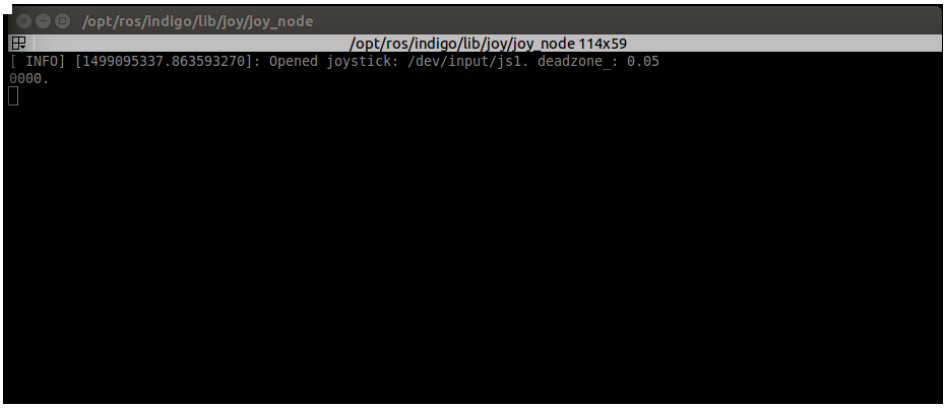

Figura 5: Tela de inicialização do comando de teleoperação por controle de Xbox de quad-rotor.

Foi desenvolvido um código que envia uma rotina de comandos de movimentação, gerando trajetórias para o quad-rotor, com o intuito de observar o sistema operando de maneira autônoma, sem uso dos controles apresentados anteriormente. Foi observada uma boa resposta do sistema a esse tipo de controle, possibilitando o uso de planejadores de trajetória que possam alcançar objetivos estipulados pelo usuário de maneira autônoma; a Figura 6 mostra o sistema sendo iniciado no computador.

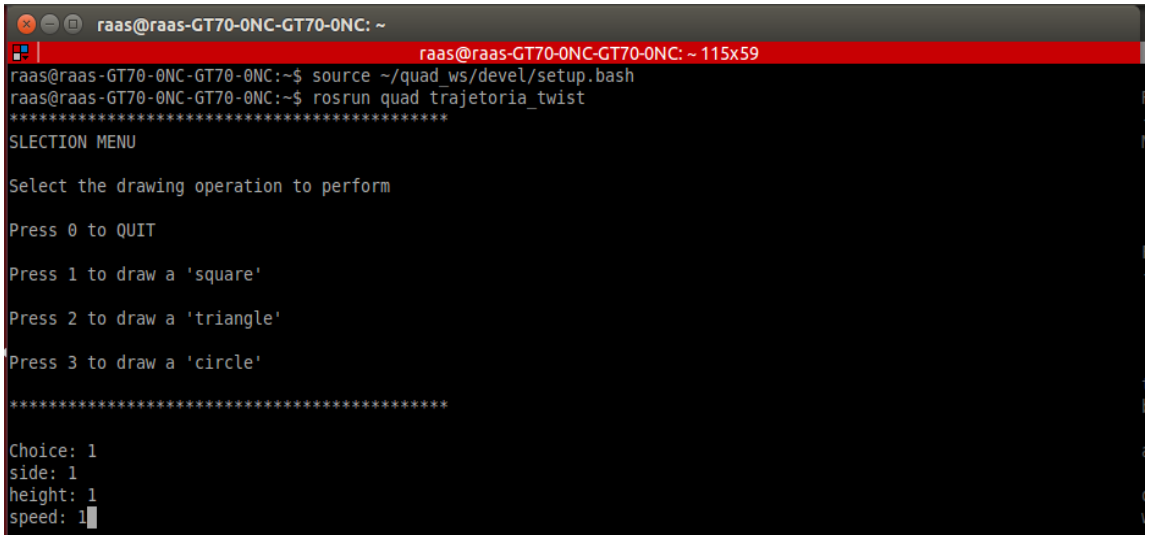

Figura 6: Tela do terminal que inicia programa gerador de trajetórias.

Utilizando a ferramenta disponível para comunicação entre o LabVIEW™ e myRIO com o sistema ROS, foi elaborado um programa que comunicasse as partes. Não foram obtidos os resultados esperados ao utilizar a ferramenta, sendo possível apenas comunicar um computador utilizando LabVIEW™ ao sistema ROS, não sendo possível a conexão entre o myRIO e ROS; o programa utilizado no computador e myRIO pode ser observado nas Figuras 7 e 8.

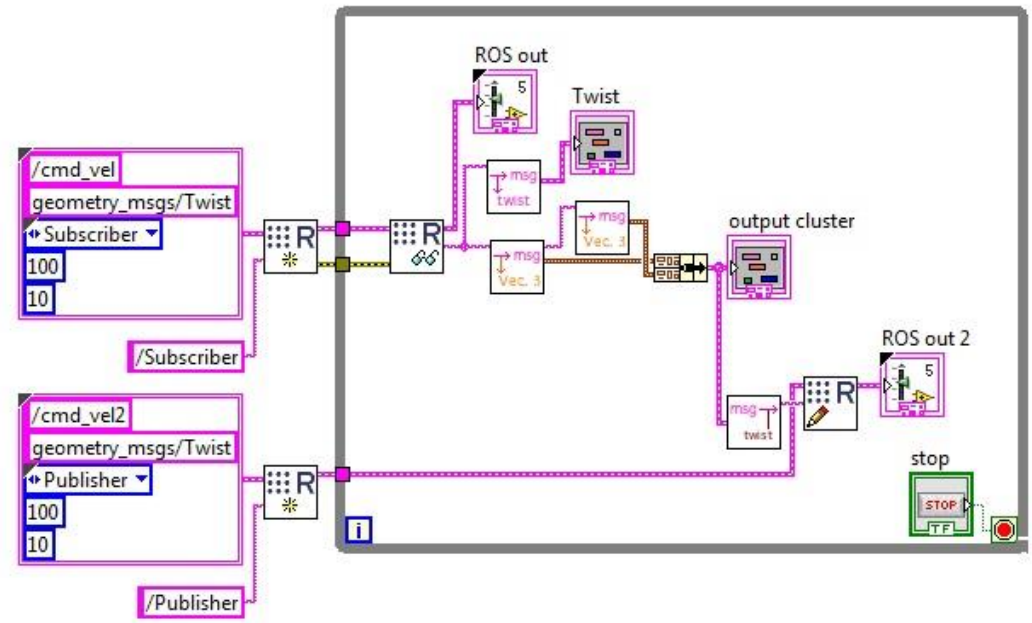

Figura 7: Programa em linguagem de blocos para comunicação entre LabVIEW™ e ROS.

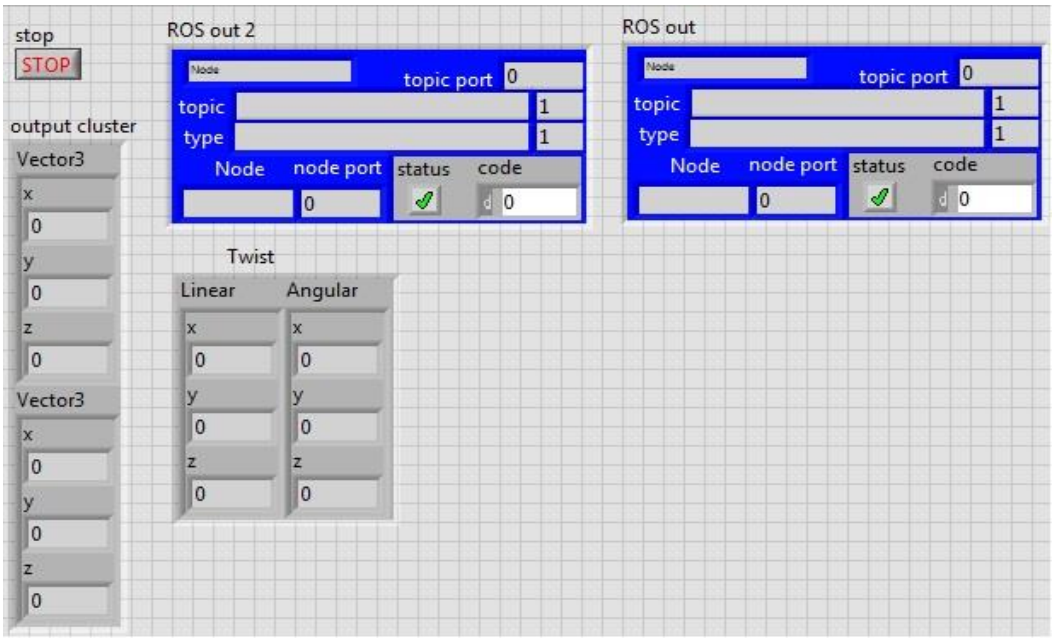

Figura 8: Painel frontal do programa em LabVIEW™ que se comunica com ROS.

Após identificar problemas de comunicação para os quais não foi obtido sucesso em solucioná-los utilizando a plataforma myRIO e LabVIEW, foi proposta como solução para integração a plataforma de prototipagem Arduino com diferentes versões, porém, não foi obtido o resultado esperado, novamente. Foi identificada a falta de compatibilidade da versão Due, por ser uma plataforma com arquitetura interna diferente da maior parte das placas e baixa eficiência do sistema na versão Duemilanove. Não foi possível estabelecer uma comunicação que atendesse às necessidades deste trabalho.

Discussão e conclusão

Durante o processo de desenvolvimento do sistema, foi observada uma grande dificuldade na reutilização do código ROS de terceiros, devido à falta de suporte. Os pacotes de ROS tinham pouca documentação e elevado nível de conhecimento no framework como pré-requisito para o uso. Foi necessária a dedicação da maior parte do tempo disponível para elaboração do trabalho para entender o funcionamento dos pacotes e desenvolver adaptações para o projeto.

Foi necessário identificar por quais caminhos os dados deveriam ser distribuídos e quais seus formatos. Diversos pacotes foram analisados para a escolha final do hector_quadrotor como base para o sistema deste trabalho, devido a uma maior documentação disponível e muitos usuários da comunidade ROS.

Apesar de o pacote hector_quadrotor ter aplicação em sistemas apenas simulados, a integração com um sistema real se mostrou possível, evidenciando a utilidade do pacote para potencializar a concepção de quad-rotores e tecnologias utilizadas nesse tipo de robôs. Através da análise do fluxo dos dados pelos tópicos e a estrutura do sistema pode-se oferecer um sistema de visualização e telemetria para o sistema real e simulação.

Foi necessária a alteração de arquivos de inicialização para carregamento de funcionalidades diferentes do padrão (teleoperação e ambiente de simulação simplificado). Outro desafio foi a integração do sistema ROS que é prioritariamente utilizado em plataforma Linux e o myRIO, que é utilizado principalmente em plataforma Windows. Foi necessária a utilização de um pacote de comunicação do sistema ROS com programas que não são do mesmo framework. Ao mesmo tempo, foi necessário o uso de uma ferramenta no LabVIEW que permitia a comunicação e programação que atendessem ao framework ROS.

Identificamos que a ferramenta disponível para comunicação entre o LabVIEW e myRIO com o sistema ROS não tem mais suporte do desenvolvedor o que dificulta a sua utilização. Não foram obtidos os resultados esperados ao utilizar a ferramenta, sendo possível apenas comunicar um computador utilizando LabVIEW ao sistema ROS.

Como solução para integração, foi utilizada a plataforma de prototipagem Arduino, com diferentes microcontroladores, porém não foi obtido o resultado esperado. Devido à falta de compatibilidade ou baixa eficiência do sistema, não foi possível estabelecer uma comunicação que atendesse às necessidades deste trabalho. Não foi possível realizar a integração entre um quad-rotor genérico e o sistema ROS, devido à falta de tempo para contornar os desafios encontrados com a comunicação entre microcontrolador e sistema ROS.

Foi possível chegar à conclusão, através do desenvolvimento do sistema, que é possível integrar um quad-rotor genérico ao sistema ROS utilizando uma abordagem diferente do que é encontrado na comunidade, utilizando informações do sistema real para alimentar o ambiente virtual e utilizando menos sistemas proprietários e mais aplicações desenvolvidas pelo grupo de trabalho.

O progresso desarticulado de soluções para o sistema foi muito vantajoso, pois possibilitou trabalhar em diferentes partes do projeto sem prejudicar outras. Foi possível gerar códigos com diferentes funcionalidades que, posteriormente, foram integradas ao sistema, devido à metodologia do framework ROS.

Foram aproveitados códigos de terceiros, disponibilizados pela comunidade. Empregar porções ou integralmente esses códigos não foi uma tarefa trivial, devido à dificuldade de interpretar e usar códigos com nenhuma ou pouca documentação clara, porém, mostrou-se uma grande vantagem utilizá-los, após dominar a metodologia do framework e a grande quantidade disponível para adaptação.

Foi empregado o ambiente de simulação do pacote hector_quadrotor, que se mostrou adequado para o projeto. Foi possível, nesse ambiente, testar um simulador de trajetórias e sistemas de teleoperação. Quando integrado com o microcontrolador Arduino duemilanove, funcionou de maneira pouco satisfatória, devido às distorções de trajetória oriundas dos atrasos na comunicação.

Foram utilizadas no projeto as plataformas Linux (Ubuntu), Windows 7, LabVIEW 14, Arduino, myRIO e ROS. A utilização das diversas plataformas se mostrou uma grande vantagem do framework, possibilitando a utilização de diversas ferramentas e possível integração entre elas. Foi identificado que as ferramentas disponíveis para utilizar o LabVIEW e myRIO não apresentam suporte do desenvolvedor e não apresentaram boas perspectivas de uso futuro. As outras plataformas têm diversas aplicações disponíveis e boa compatibilidade.

Pode-se observar uma boa aplicação dos pacotes de teleoperação e utilizá-los torna o sistema mais relevante, oferecendo a possibilidade de utilizar diversos meios para gerar comandos de movimentação através de teclado, joystick e rotinas geradas por códigos que geram uma sequência de comandos no sistema.

Foi alcançado o objetivo de criar um sistema operacional para quad-rotores genéricos, utilizando porções ou integralmente códigos disponibilizados pela comunidade e adicionando códigos gerados pelo autor. Após vencida a barreira de interpretação dos códigos de terceiros e do método oferecido pelo framework, foi observado que esse ambiente pode facilitar as pesquisas futuras com quad-rotores, dando um direcionamento para evolução de novas aplicações para quad-rotores.

Devido à limitação de tempo, não foi possível uma integração com um protótipo de quad-rotor genérico real, mas é uma possibilidade plausível que, após a solução de problemas de comunicação entre a plataforma de microcontrolador utilizada e o sistema ROS, poderá ser alcançada.

Entre os programas de computador utilizados apenas o LabVIEW não era open source. Porém, foi utilizada a licença que é oferecida junto ao myRIO. Logo, não foi gerado custo extra. Os equipamentos utilizados já estavam disponíveis, o que fez o projeto não ter gerado ônus financeiro, mostrando que o progresso tem baixo investimento inicial e uma perspectiva de baixo orçamento necessário para futuros projetos que derem prosseguimento a esse sistema.

Diante disso, conclui-se que o desenvolvimento do sistema operacional ROS para quad-rotores genéricos proposto por este trabalho necessita de um extenso período de aprendizado e analise de códigos para reutilização. Vencida essa etapa, reduz-se o tempo

necessário para projetar e executar novos protótipos e aplicações de quad-rotores indicando que sua aplicação é ideal quando é prevista a continuação do trabalho, desta forma, diminui-se o tempo de desenvolvimento. Introduz custo mínimo para o projeto viabilizando a utilização deste e podendo potencializar o estudo de VANT's, oferecendo um ambiente desarticulado de concepção, porém, com integração amigável através do framework ROS.

Referências

ALEJO, D. *et al.* **Optimal Reciprocal Collision Avoidance with mobile and static obstacles for multi-UAV systems**. 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings, p. 1259–1266, 2014.

BOUABDALLAH, S.; MURRIERI, P.; SIEGWART, R. **Design and control of an indoor micro quadrotor**. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, v. 5, n. April, p. 4393–4398, 2004.

BRESCIANI, T. Modelling **, Identification and Control of a Quadrotor Helicopter.** English, v. 4, n. October, p. 213, 2008.

COUSINS, S. *et al.* **Sharing software with ROS**. IEEE Robotics and Automation Magazine, v. 17, n. 2, p. 12–14, 2010.

DEMARCO, K.; WEST, M. E.; COLLINS, T. R. **An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV).** Oceans 2011, p. 1–7, 2011.

DUNKLEY, O. *et al.* **Visual-Inertial Navigation for a Camera-Equipped 25 g Nano-Quadrotor**. IROS2014 Aerial Open Source Robotics Workshop, p. 4–5, 2014.

GAZEBO. Disponível em: www.gazebosim.org. Acesso em: 02 de maio de 2016.

GRABE, V. *et al.* **The TeleKyb *framework* for a modular and extendible ROS-based quadrotor control**. 2013 European Conference on Mobile Robots, ECMR 2013 - Conference Proceedings, p. 19–25, 2013.

MARTINEZ, A.; FERNÁNDEZ, E. **Learning ROS for Robotics Programming**.Packt Publishing 2013.

MELO, SALLES E ALMEIDA, 2010. **Implementação De Uma Aeronave Miniatura Semiautônoma Com Quatro Propulsores Como Plataforma De Desenvolvimento**. XVIII Congresso Brasileiro de Automática - CBA, p. 1805 – 1810, 2010.

MEYER, J. *et al.* **Comprehensive simulation of quadrotor UAVs using ROS and Gazebo**. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 7628 LNAI, p. 400–411, 2012.

O'KANE, Jason M. **A Gentle Introduction to ROS**. 2014.

ROS, **About ROS. 2015**. Disponível em: http://www.ros.org/about-ros/. Última visita: 09 jan 2016.

DEVELOPMENT OF AN OPERATIONAL SYSTEM ROS FOR QUADCOPTERS

ABSTRACT

The study of robots has become recurrent in the academic environment, in professional and hobbyist sectors, this is leveraged by component costs decreasing and the availability of free tools such as: ROS; Arduino; and Linux. One of the greatest obstacles of the robotic system development task is the integration of the several constructive components, these manufactured by different companies and with diverse methodologies of use. This work aims at the elaboration of an operating system using the development framework ROS (Robot Operating System), which proposes the integration's improvement of subsystems and components belonging to robots. This environment favors design on several concomitant fronts, maintaining the system's cohesion. Another key factor is that ROS is a free environment with a large collaborative community, offering several modules of specific functionalities (e.g. remote operation, position control) and examples available for use. The main contribution of this work is the elaboration of a system that can aid in the expansion of technologies applied to quad-rotor type robots, allowing faster advances and in harmony with what has been used by other researchers.

Keywords*: Robot systems development. Quadcopter. ROS. Quadcopter simulator. Telemetry*

Enviado em: 10/2017.
Aceito em 01/2018.