Estudo e análise de sensores para Veículos Aéreos Não-Tripulados

Study and analysis of sensors for Unmanned Aircraft Vehicle

Aroldo Katsuyoshi Simada Komori *, Marco Aurélio Wehrmeister†

RESUMO

Soluções baseadas em Veículos Aéreos Não-Tripulados (VANT) tem sido aplicadas em várias áreas distintas devido sua mobilidade e alcance as áreas de difícil acesso, principalmente em serviços nos setores de segurança e logística. Diante desse contexto, a garantia de um sistema de controle e integração de sensores se torna um fator importante, visto que quedas ou perda de controle de um VANT podem causar graves problemas. Na integração de sensores de robôs, incluindo de VANTs, o Robot Operating System (ROS) possui um papel fundamental na comunicação e interfaceamento dos sensores embarcados. Esse projeto visa o estudo e análise do comportamento do sistema de controle aéreo de código aberto PX4 Autopilot junto ao ROS utilizando um sensor LIDAR e um sensor de profundidade em ambiente simulado.

Palavras-chave: Robot Operating System. VANT. PX4 Autopilot.

ABSTRACT

Solutions based on Unmanned Aerial Vehicles (UAV) have been applied in several different areas due to their mobility and reach to areas of difficult access, especially in services in the security and logistics sectors. In this context, the assurance of a control system and sensor integration becomes an important factor, since crashes or loss of control of a UAV can cause serious problems. In the integration of robot sensors, including UAVs, the Robot Operating System (ROS) has a fundamental role in the communication and interfacing of on-board sensors. This project aims to study and analyze the behavior of the open source PX4 Autopilot air control system with the ROS using a LIDAR sensor and a depth sensor in a simulated environment.

Keywords: Robot Operating System. UAV. PX4 Autopilot.

1 INTRODUÇÃO

A utilização de VANTs em grandes projetos de logística, de inspeções em áreas de risco ou difícil acesso, têm sido de grande destaque e está cada vez mais próximo do cotidiano das pessoas. Seja qual for a aplicação atribuída, o sistema de sensoriamento e controle de voo são fatores imprescindíveis para a estabilidade do VANT e, consequentemente, para o sucesso da tarefa a ser executada.

Na área de robótica, a comunicação dos dados dos sensores é feita usualmente através do Robot Operating System (ROS)(ROS, s.d.[a]), que é um framework de código aberto com ferramentas e bibliotecas que auxiliam na comunicação entre unidades de processamento e seus dispositivos de entrada e saída, como os sensores. Em termos de sistema de controle de voo, o PX4 Autopilot(PX4, s.d.[b]) é um firmware de código aberto utilizado no desenvolvimento de VANTs e com recursos de suporte ao ROS tanto para o ambiente real como simulado.

* Laboratório de Engenharia de Sistemas Computacionais (LESC), Bacharelado em Engenharia Mecatrônica; aroldok@alunos.utfpr.edu.br .

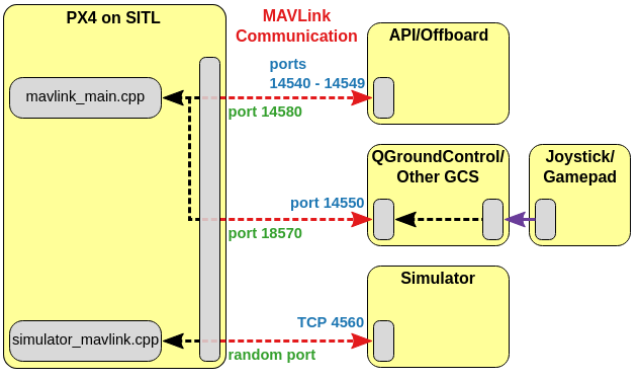
† Laboratório de Engenharia de Sistemas Computacionais (LESC), Programa de Pós-Graduação em Engenharia Elétrica de Informática Industrial (CPGEI); wehrmeister@utfpr.edu.br; https://orcid.org/0000-0002-1415-5527.

Devido às restrições impostas pelas ações de combate a pandemia COVID-19, o presente projeto é restrito ao estudo e análise dos recursos de simulação do PX4 SITL (software in the loop) incluindo o sensor LIDAR e o sensor de profundidade embarcado. Através do pacote mavros(ROS, s.d.[c]) analisou-se a comunicação dos nós no ROS com o protocolo MAVLink(MAVLINK, s.d.) no simulador de robótica 3D Gazebo(GAZEBO, s.d.).

2 AMBIENTE DE EXECUÇÃO

O ambiente de execução utilizado para os experimentos com os sensores foi composto de um computador desktop com a seguinte configuração: intel core i7(8th Gen), 20GB RAM, NVIDIA GeForce MX110, sistema operacional Ubuntu 20.04 LTS Focal Fossa, distribuição ROS Noetic Ninjemys, simulador Gazebo 11, sistema de controle aéreo PX4 SITL v1.12.1. A figura 1 apresenta uma visão geral da arquitetura de comunicação utilizadas neste trabalho.

Figura 1 – Comunicação do PX4 em ambiente simulado

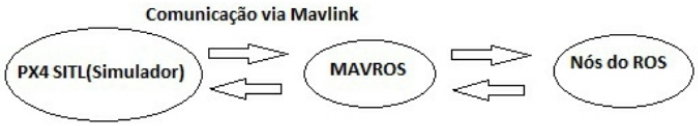


Fonte: PX4 User Guide

O pacote utilizado na integração do ROS com o sistema PX4 foi o mavros, juntamente ao modelo Quad Iris(PX4, s.d.[a]) fornecido pelo PX4 e os sensores 2D-LiDAR e de profundidade com compatibilidade oficial com o ROS em formato URDF (Unified Robot Description Format)(ROS, s.d.[g]).

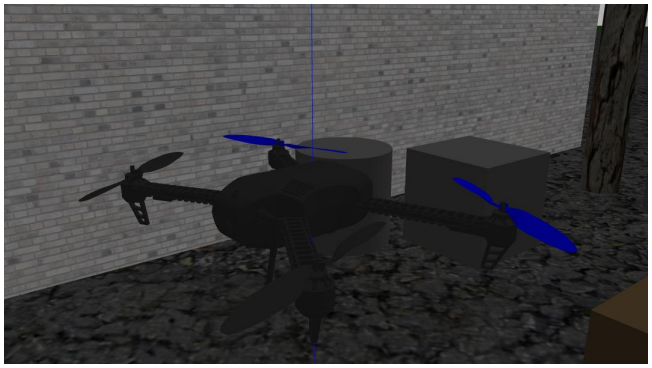
A figura 2 mostra o papel do pacote mavros na comunicação entre o ROS e o sistema do PX4 do qual fornece um driver de comunicação para o sistema do PX4 que utiliza o protocolo mavlink. Já a figura 3 mostra o modelo Quad Iris cujo modelo base é implementado de forma padrão junto a instalação do sistema PX4.

Figura 2 – Função do MAVROS



Fonte: Autoria própria (2021)

Figura 3 – Modelo Quad Iris.

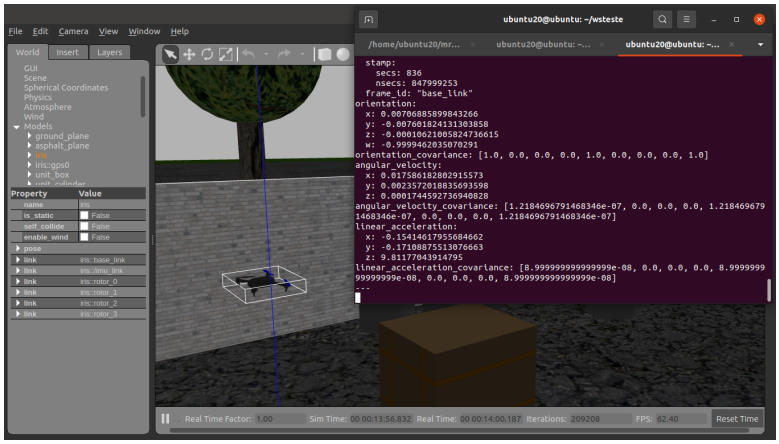


Fonte: Autoria própria (2021)

3 RESULTADOS

Após a instalação do ambiente de execução foi realizado um teste de decolagem cujo comando foi enviado via rosservice provido pelo pacote mavros. Foi possível obter os parâmetros do sistema inercial de navegação do Quad Iris com a decolagem já sucedida. A figura 4 apresenta o conteúdo do tópico com esses parâmetros.

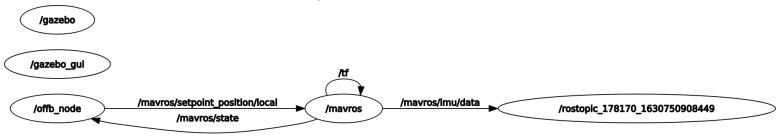
Figura 4 – Leitura do IMU do Quad Iris



Fonte: Autoria própria (2021)

Utilizando o framework de visualização gráfica do ROS chamado rqt(ROS, s.d.[e]), é possível observar a relação de nós e tópicos estabelecida pelo comando de decolagem e a publicação do tópico contendo o imu do Quad Iris, permitindo também a checagem de possíveis erros. A figura 5 apresenta a vizualização do rqt para o caso estudado, onde é possível observar o mavros recebendo informações referentes ao posicionamento via Mavlink e transmitindo os dados do imu ao nó publicador.

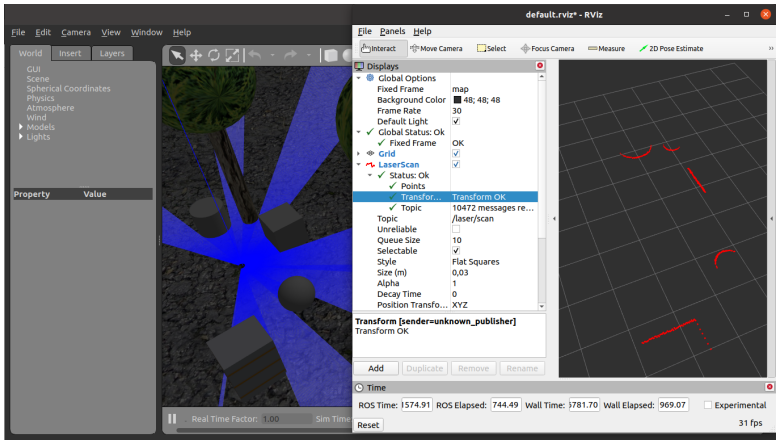
Figura 5 – Visualização de nós e tópicos com rqt_graph



Fonte: Autoria própria (2021)

A instalação do sensor 2D-LiDAR no VANT virutal foi realizada acoplando o sensor no formato URDF no modelo do Quad Iris. Após a implementação do sensor, foi colocado no ambiente objetos aleatórios para o teste de leitura. Na visualização da nuvem de pontos gerada foi utilizada a ferramenta de inspeção 3D do ROS chamada rviz(ROS, s.d.[f]) que possibilita a visão gráfica da leitura de tópicos. O tópico visualizado foi o LaserScan, geralmente utilizado por nós na publicação da leitura de sensores laser planares. A figura 6 mostra, ao lado esquerdo, o Quad Iris com os feixes do LIDAR direcionado a objetos colocados aleatoriamente no ambiente. Ao lado direito, é possível vizualizar o tópico LaserScan que contém os dados da leitura do sensor pelo rviz.

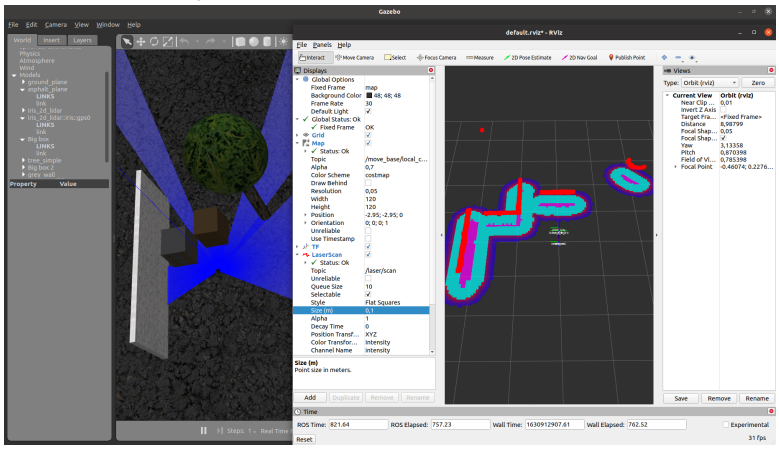
Figura 6 – Teste e vizualização das leituras do 2D-LiDAR



Fonte: Autoria própria (2021)

Com a implementação do 2D-LiDAR, foi testado o pacote costmap_2d(ROS, s.d.[b]) que é utilizado para mapeamento de áreas de navegação e como parâmetros para desvio de obstáculos na trajetória de robôs. Com esse pacote, é possível a geração de um costmap do qual trata-se basicamente de um mapa com malhas onde possuem parâmetros que são utilizados por algoritmos de planejamento de trajetória tais como o Dijkstra e o A*(A-estrela). O estudo se restringiu a implementação e visualização dos tópicos por meio da ferramenta rviz. A figura 7 mostra a leitura do tópico do pacote costmap_2d junto ao LaserScan onde é possível visualizar a geração do costmap a partir da detecção de obstáculos pelo LIDAR.

Figura 7 – Formação do costmap utilizando com as leituras do 2D-LiDAR

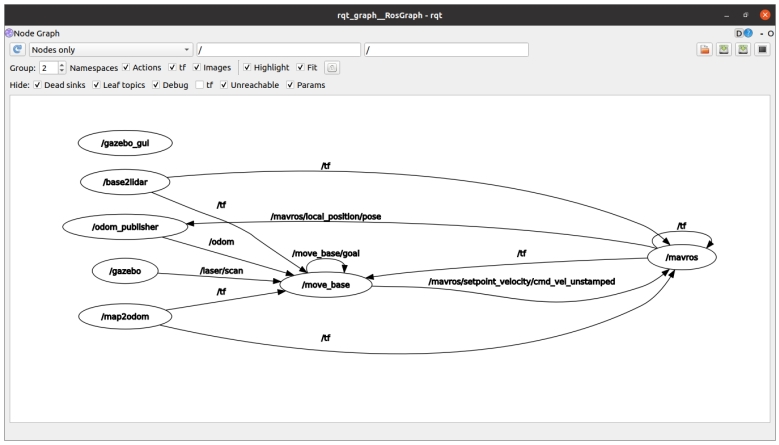


Fonte: Autoria própria (2021)

Utilizando o rqt_graph foi também observada a relação de nós e tópicos do ROS na situação estudada, onde é

possível ver a transmissão da leitura do LIDAR e da odometria do Quad Iris tanto para o mavros quanto para o nó base para utilização do costmap. O nó de geração do costmap por sua vez, utiliza os tópicos de rastreagem das transformações de frames oriundos do mavros e o nó base citados anteriormente. Tal relação vista utilizando a ferramenta rqt_graph é apresentada na figura 8.

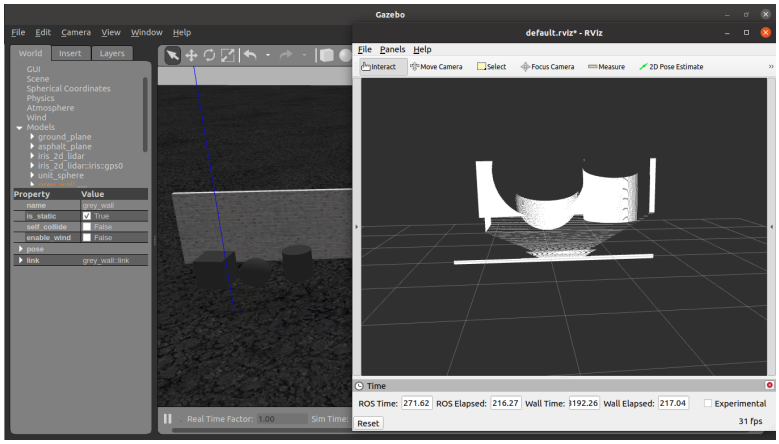
Figura 8 – Vizualização dos nós e tópicos envolvidos com rqt



Fonte: Autoria própria (2021)

Finalmente, avaliou-se o sensor de profundidade começando pela implementação de uma câmera RGB-D(ROS, s.d.[d]) no modelo Quad Iris e a visualização de tópicos de leitura no rviz da nuvem de pontos gerada. A figura 9 mostra a visualização da nuvem de pontos coletada pela câmera RGB-D pelo tópico DepthCloud onde é possível verificar a detecção de objetos.

Figura 9 – Vizualização gráfica da nuvem de pontos gerada



Fonte: Autoria própria (2021)

Com os testes realizados utilizando o 2D-LiDAR e a câmera de profundidade, foi possível o estudo da utilização do sistema PX4 além das leituras dos sensores dos quais se tornam bases para implementação de sistemas de navegação e mapeamento utilizando robôs, assim como VANTs. Apesar da ausência das dificuldades comumente enfrentadas em testes na realidade tais como em etapas de montagem do VANT e de sensores a serem acoplados, como também na avaliação da estabilidade de voo em condições reais, durante as atividades remotas surgiram diversos problemas envolvendo principalmente a compatibilidade da distribuição ROS utilizada, Noetic Ninjemys em relação a pacotes e dependências das quais serviam apenas a distribuições mais antigas tais

como o Kinetic Kame e Melodic Morenia, sendo necessário por vezes alterações em scripts de instalação e a implementação de dependências a parte.

4 CONCLUSÕES

Pela realização do estudo e análise de sensores embarcados em VANTs foi possível compreender o processo de implementação e leitura de sensores LiDAR e da câmera de profundidade utilizando ferramentas de análises e pacotes do ROS. O estudo se limitou a testes básicos em ambiente simulado mas que se tornou grande aprendizado para um iniciante sobre os recursos apresentados. Infelizmente, devido as restrições impostas pela pandemia em relação as atividades presenciais no período da bolsa, não foi possível seguir a etapa de testes com sensores reais em laboratório. Tais atividades se mostram necessários tendo em vista a futura realização de ensaios envolvendo a comparação da leitura de sensores reais ou até de projetos com a integração das leituras do sensor LIDAR e da câmera de profundidade estudados em algoritmos de visão computacional ou aprendizado de máquinas.

AGRADECIMENTOS

Os autores agradeçem o recebimento da bolsa fomentada pela Universidade Tecnológica Federal do Paraná, como também os membros do Laboratório de Engenharia de Sistemas Computacionais da UTFPR.

REFERÊNCIAS

GAZEBO. **Gazebo Tutorials.** Disponível em : <http://gazebosim.org/tutorials>. Acesso em: 23 ago. 2021.

MAVLINK. **Mavlink Developer Guide.** Disponível em : <https://mavlink.io/en/>. Acesso em: 23 ago. 2021.

PX4. **Gazebo Vehicles.** Disponível em : <https://docs.px4.io/master/en/simulation/gazebo_vehicles.html>. Acesso em: 23 ago. 2021.

PX4. **PX4 User Guide.** Disponível em : <https://docs.px4.io/master/en/>. Acesso em: 23 ago. 2021.

ROS. **About ROS.** Disponível em : <https://www.ros.org/about-ros/>. Acesso em: 23 ago. 2021.

ROS. **costmap_2d.** Disponível em : <hhttp://wiki.ros.org/costmap_2d>. Acesso em: 23 ago. 2021.

ROS. **mavros.** Disponível em : <http://wiki.ros.org/mavros>. Acesso em: 23 ago. 2021.

ROS. **rgbd_launch.** Disponível em : <http://wiki.ros.org/rgbd_launch>. Acesso em: 23 ago. 2021.

ROS. **rqt.** Disponível em : <http://wiki.ros.org/rqt>. Acesso em: 23 ago. 2021.

ROS. **rviz.** Disponível em : <http://wiki.ros.org/rviz>. Acesso em: 23 ago. 2021.

ROS. **urdf.** Disponível em : <http://wiki.ros.org/urdf>. Acesso em: 23 ago. 2021.