INTERFACEAMENTO E CONTROLE DE VEÍCULOS AÉREOS NÃO TRIPULADOS

André Ruschel [1], Lucas Schulze[2], Douglas Wildgrube Bertol[3]

[1] Acadêmico do Curso de Engenharia Elétrica – DEE – bolsista PROBIC/UDESC
[2] Acadêmico do Curso de Engenharia Elétrica – DEE – voluntário
[3] Orientador, Departamento Engenharia Elétrica – douglas.bertol@udesc.br

Palavras-chave: Veículo Aéreo Não Tripulado. Arquitetura de *Hardware*. Arquitetura de *Software*. Interface de comando.

Um veículo aéreo não tripulado designado a realizar atividades de forma autônomas, requer um controlador de resposta rápida, robusto e tolerante a falhas. Inclusive, em muitas das tarefas executadas por ele, são necessárias uma grande carga de processamento computacional, pois também são executados, paralelamente, computações como o processamento de imagens. Para atender a estes requisitos de controle e processamento, foi proposta uma arquitetura de *hardware* envolvendo duas plataformas microcontroladas: a Pixhawk e o kit Raspberry Pi/Navio2.

A pesquisa realizada foi de natureza aplicada, com uma abordagem exploratória e levantamento bibliográfico através do estudo do estado da arte publicado em artigos das áreas de controle de veículos aéreos móveis e sistemas de tempo real. Além da busca de soluções comerciais existentes, encontradas em sítios da internet.

Primeiramente, no projeto foram desenvolvidos testes para validação da utilização da plataforma de *hardware* conhecida por Navio2, que é considerada um *Hardware Attached on Top* (HAT) e produzida e distribuída pela empresa Emlid. Ela é desenvolvida para funcionar em conjunto (acoplada) ao microcontrolador Raspberry Pi, fornecendo os instrumentos necessários para que o conjunto possa controlar VANTs.

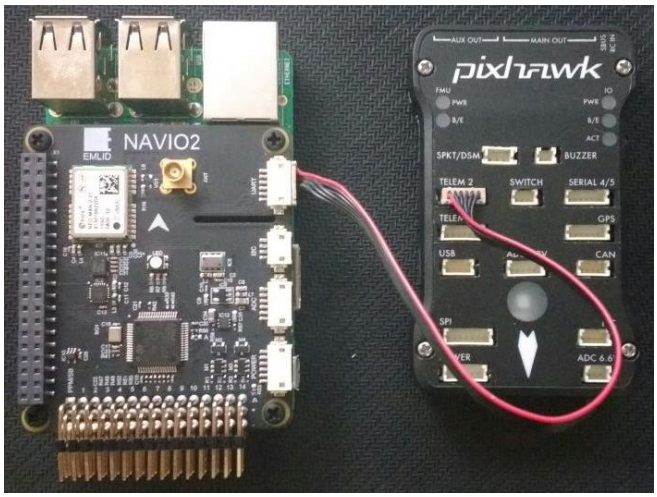
Desta primeira etapa do projeto, foram lançados esforços para o desenvolvimento de uma imagem do sistema operacional Linux customizada para o kit da Navio2, pois a imagem fornecida pela fabricante Emlid continha algumas falhas e aplicações desnecessárias para o objetivo da pesquisa. O objetivo desta imagem era a conversão do *kernel* do sistema operacional para uma versão com características de *soft real time*. Tal conversão permitiria o desenvolvimento de aplicações no campo da robótica de processos que requerem *deadlines* predefinidos, assim sendo possível identificar potenciais falhas em voo e a execução de controladores em abstrações superiores de *softwar*e.

Também foi avaliada a utilização da plataforma *Robot Operating System* (ROS), para gerência e efetivação da comunicação entre diversos módulos de *hardware* e *software* que compõe a arquitetura geral dos VANTs utilizados no projeto.

Os resultados desta primeira etapa levaram apenas a ações de adequação dos objetivos do projeto, gerando a reavaliação da utilização da Navio2 como única solução de *hardware*. As maiores dificuldades encontradas foram devido à escassez e obsolescência das informações sobre o kit e até mesmo algumas incompatibilidades do microcontrolador Raspberry Pi com a utilização de *kernels* de tempo real. Além disso, o fabricante Emlid não fornece nenhum meio para que criássemos a comunicação da placa Navio2 e o microcomputador. Em janeiro de 2017 a Emlid forneceu, em seu site, uma imagem com a plataforma ROS já instalado.

Na segunda etapa da pesquisa, foi optado pela utilização plataforma de *hardware* Pixhawk. Os primeiros esforços realizados foram no sentido de conhecer a pilha de *software* da solução e estabelecer a comunicação de dados sensoriais entre a o kit Raspeberry Pi/Navio2 e a Pixhawk como é mostrado na Figura 1. Esta comunicação viabiliza o desenvolvimento de soluções para aumentar a robustez dos dados de voo e também a implementação de uma arquitetura com características de *fail-safe*, para o caso de um dos controladores falharem (situação crítica em voo).

Figura 1 – Plataformas de *hardware* e a ligação para comunicação (serial) entre elas.



Fonte: próprio autor.

Foi desenvolvida a comunicação serial (UART) entre os módulos, utilizando o protocolo de comunicação *mavlink*. Na Pixhawk o firmware utilizado é do *flight stack* ArduPilot. Durante a montagem da solução, ocorreram algumas falhas na comunicação quando utilizado as entradas UART. Tal problema foi contornado provisoriamente, utilizado a porta USB.

Os trabalhos atuais e futuros desta pesquisa estão focados na finalização da arquitetura de *software*, possibilitando a implementação de algoritmos com requisitos de tempo real, como, por exemplo, soluções de fusão de sensores utilizando algoritmos de filtragem de Kalman, objetivando principalmente a robustez dos dados de voo.