TRABALHO DE GRADUAÇÃO 1

TÍTULO DO TRABALHO DIVIDIDO EM MAIS DE UMA LINHA PARA TÍTULOS REALMENTE LONGOS COMO ESTE

Eduardo Moura Cirilo Rocha

Brasília, Junho de 2017

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO 1

TÍTULO DO TRABALHO DIVIDIDO EM MAIS DE UMA LINHA PARA TÍTULOS REALMENTE LONGOS COMO ESTE

Eduardo Moura Cirilo Rocha

Orientador: Prof. Henrique Cezar Ferreira, ENE/UNB

Brasília, Junho de 2017

SUMÁRIO

Capítulo 1

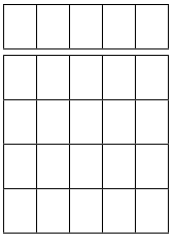
Introdução

Resumo opcional

1.1 Contextualização

Contextualizar.

Conforme [**?**], vide a Tabela 1.1. Assim sendo, observe a Figura 1.1.



Descrever tabela.

1.2 Definição do problema

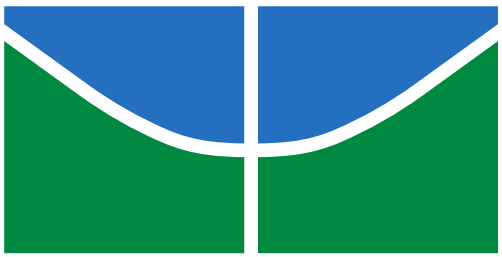
Definir problema.

1.3 Objetivos do projeto

Objetivos.

1.4 Resultados obtidos

Resultados.



Descrever figura.

1.5 Apresentação do manuscrito

Apresentar.

Capítulo 2

Descrição do Sistema

2.1 Introdução

O sistema completo é composto de quatro sub-sistemas principais, três aeronaves, entre elas uma asa voadora, e uma estação base. A comunicação entre eles se dá pelo uso de rádios de comunicação. Cada sub-sistema possui seu próprio modem de comunicação e tem a capacidade de comunicar-se diretamente com qualquer outro sub-sistema.

Cada aeronave possui dois componentes principais que valem ser mencionados. Um deles é o piloto automático, que é responsável pela aquisição, condicionamento e processamento de sinais provenientes dos sensores da aeronave e pelo controle dos atuadores do avião. O outro é o computador embarcado. Ele é responsável pelo processamento de dados, controle dos aviões (controle individual de cada avião ou controle cooperativo entre os três aviões) e pela comunicação com os outros sub-sistemas por meio dos rádios de comunicação.

Esse capítulo descreve e detalha os componentes do sistema e a comunicação entre eles. colocar foto aqui

2.2 Aeronaves

Utilizou-se dois aviões do modelo *X-UAV Skua FPV Plane*, onde *FPV* é a abreviação de *first-person view*. Isso significa que esse avião pode ser controlado ou pilotado remotamente, pelo uso de câmeras. Esse avião possui uma envergadura grande, de $2,1$ metros, e é relativamente leve [1]. Essas características resultam em um avião de excelente performance energética e de baixa velocidade. Um dos aviões desse modelo pode se visto na figura X.

A asa voadora utilizada é do modelo *Go Discover FPV Plane*, que também é do tipo *FPV*. Ela possui envergadura de $1,6$ metros e também é relativamente leve [2]. Em relação aos aviões do modelo *X-UAV Skua FPV Plane*, a asa voadora é mais ágil, porém possui pior performance energética. O avião desse modelo pode se visto na figura X.

2.3 Piloto automático - Pixhawk

2.3.1 Introdução

Escolheu-se usar o Pixhawk como o piloto automático de cada avião. Pixhawk é um projeto independente de *hardware* aberto com o objetivo de proporcionar um piloto automático a baixo custo e de alto desempenho. Dentre as opções de pilotos automáticos disponíveis no mercado, decidiu-se que o Pixhawk tem o melhor custo-benefício para o projeto.

O Pixhawk é uma plataforma completa de *hardware* e *software*, bem como um computador, e pode executar várias aplicações de alto nível. Ele oferece um ambiente de programação compatível com sistemas Unix / Linux, o que facilita o desenvolvimento de aplicações de *software* que rodem nele. O sistema Pixhawk possui capacidade de *multithreading*, ou seja, pode executar várias tarefas simultaneamente sem que uma interfira na outra através do compartilhamento de recursos do processo. Além disso, ele possui funções de piloto automático integrado com logs detalhados de missões e comportamento de voo.

Algumas das características mais importantes do Pixhawk são [3]:

- Processador avançado Cortex® ARM 32 bits rodando NuttX RTOS;
- 14 saídas PWM / servo;
- Opções de conectividade para periféricos adicionais (UART, I2C, CAN);
- Fontes de alimentação redundantes e *failover* automático;
- Cartão microSD que permite gravação de longos logs de voo.

Dimensões do Pixhawk [3]: Peso: 38g; Largura: 50mm; Espessura: 15.5mm; Comprimento: 81.5mm.

colocar foto aqui

A preparação do Pixhawk para voo consiste principalmente em duas etapas, instalação do *firmware* no dispositivo e a calibração dos sensores ligados a ele. As próximas seções descrevem em detalhe essas duas etapas. Além disso, deve-se configurar o modo de segurança *Failsafe*, que define o comportamento da aeronave com a perda de comunicação com a estação base, e deve-se configurar o modo de armação do motor, que garante que o motor nunca se ligará quando o avião não estiver pronto para voo. A documentação oficial do Pixhawk pode ser encontrada no link `http://ardupilot.org/ardupilot/index.html#`.

2.3.2 Instalação de firmware e desenvolvimento de aplicações

A instalação do *firmware* no dispositivo pode ocorrer de duas formas, pelo uso de um programa de Estação de Controle Terrestre (ECT) ou diretamente pelo uso de ferramentas de desenvolvedor, sem o uso de um programa auxiliar. Uma ECT é uma aplicação de *software* que roda em um

computador de uma estação terrestre e que se comunica com um VANT pelo uso de telemetria sem fio. Ela é capaz de mostrar os dados de performance e posição provindos do piloto automático em tempo-real e pode ser usada para controlar o VANT em voo, mandando comandos para o piloto automático [3].

As principais ECT's disponíveis são Mission Planner, APM Planner 2, MAVProxy, QGroundControl e UgCS. Neste projeto, decidiu-se utilizar a ECT QGroundControl pelo fato de ser a única ECT disponível em todos os sistemas operacionais, Windows, Mac OS X, Linux, Android e iOS. Além disso, QGroundControl é um programa mais estável em relação aos outros e possui uma interface simples e eficiente. colocar imagem do QGroundControl

O uso de uso de uma ECT facilita muito a instalação do *firmware*. Ela cuida de todo o processo de conexão com o dispositivo, que ocorre pelo protocolo Mavlink e será explicado posteriormente nesse capítulo. As ECT's já fornecem versões atuais e estáveis de *firmware* para todos os tipos de veículos não tripulados que podem usar o Pixhawk. Porém, elas não permitem a instalação de outras versões de *firmware* que não sejam as disponibilizadas por elas. Assim, o desenvolvimento de aplicações e a alteração dos *firmwares* padrões disponibilizados exigem o uso de ferramentas de desenvolvedor para a instalação do *firmware* no Pixhawk [4].

Para a instalação do *firmware* através do QGroundControl, basta abrir a aba correspondente a tal ação, selecionar o tipo de veículo utilizado e seguir os passos fornecidos pelo próprio programa. Assim, será instalado a versão mais recente de *firmware* disponível para esse tipo de veículo [3].

2.3.2.1 Desenvolvimento de aplicações

2.3.3 Calibração de sensores

O Pixhawk já possui alguns sensores imbutidos:

- Giroscópio ST Micro L3GD20 (3 eixos, 16 bits);
- Acelerômetro e Magnetômetro ST Micro LSM303D (3 eixos 14-bits);
- Acelerômetro e Giroscópio Invensense MPU 6000 3 eixos;
- Barômetro MEAS MS5611.

Além deles, acrescentou-se ao sistema:

- Tudo de Pitot PX4 v1.0;
- Magnetômetro e GPS 3DR;
- Lidar-Lite LL-905.

A calibração dos sensores ocorre pelo uso de uma ECT. O programa QGroundControl já possui rotinas de calibração para todos esse sensores. Assim, o processo de calibração consiste de conectar

o Pixhawk ao QGroundControl, abrir a aba correspondente a calibração de sensores e seguir os passos fornecidos pelo próprio programa. colocar imagem do QGroundControl

Pode-se observar a presença redundante de sensores no sistema. Isso assegura uma maior confiabilidade nos dados obtidos. Como aeronaves podem sofrer vibrações muito fortes durante o voo, a redundância de sensores é essencial para esse projeto [3].

2.3.4 Diagramas de conexões

O sensor lidar pode ser conectado ao Pixhawk de duas formas, pelo protocolo I2C na porta I2C e por *pulse-width-modulation* (PWM) na trilha PWM. De acordo com a documentação do Pixhawk, o lidar utilizado apresenta problemas de interferência com outros dispositivos quando conectado na porta I2C. Assim, escolheu-se a conexão por PWM. Um diagrama de conexão pode ser vista na tabela 2.1 e o esquema de montagem pode ser visto na figura, onde o valor do resistor pode variar entre 200Ω e $1k\Omega$ [3]. Mais detalhes sobre a conexão podem ser encontrados em `http://ardupilot.org/copter/docs/common-rangefinder-lidarlite.html?highlight=lidar#`.

Sinal LIDAR-Lite	Sinal Pixhawk
J1	CH6 Out - V+
J2	CH6 Out - Signal (sinal interno 55)
J3	CH5 Out - Signal (sinal interno 54)
J4	
J5	
J6	Ch6 Out - Ground

Tabela 2.1: Diagrama de conexão entre o Lidar e o Pixhawk.

O Pixhawk pode ser energizado de duas formas, pela porta "power", que é o método mais comum, e pela trilha de portas de servos pelo uso de um BEC de 5V. Com a conexão do sensor lidar nas postas PWM, a energização da trilha de portas de servos com 5V torna-se necessária. Como não é recomendado energizar o Pixhawk somente por essa trilha, decidiu-se energizar o Pixhawk das duas formas, redundantemente. Assim, se a voltagem fornecida na porta "power"cair, o Pixhawk continua energizado. Formas mais avançadas de conexão podem ser encontradas em `http://ardupilot.org/copter/docs/common-powering-the-Pixhawk.html#common-powering-the-Pixhawk`, como, por exemplo, energização triplo-redundante [3].

Um diagrama geral pode ser visto na Um diagrama geral pode ser visto na figura

2.3.5 Pouso automático (sensor Lidar)

`http://ardupilot.org/copter/docs/common-rangefinder-lidarlite.html?highlight=lidar#`

2.4 Computador embarcado - Gumstix Overo Waterstorm COM

2.4.1 Introdução

Escolheu-se utilizar o computador embarcado Overo WaterSTORM acoplado a uma placa de extensão TOBI, que possui módulos auxiliares essenciais. Acoplou-se também ao sistema uma câmera Caspa VL, capaz de capturar imagens coloridas com dimensão de 752 x 480 pixels em uma frequência de 60 imagens por segundo. Esses três componentes, que podem ser vistos na figura, são produzidos pela empresa Gumstix, fabricante de *hardware* especializada em computadores pequenos do tipo computador-em-módulo (COM - *computer-on-module*), muito utilizados para sistemas embarcados.

Apesar do tamanho pequeno, a combinação da Overo COM com a placa de extensão TOBI possui o mesmo desempenho do que um computador Linux completo de tamanho normal, maior do que outros sistemas desse tipo encontrados no mercado, como, por exemplo, o computador Raspberry Pi. Pode-se ver como uma Overo COM é pequena na figura, onde o computador encontra-se ao lado de uma bateria AA.

Algumas das especificações mais importantes do Overo Waterstorm COM são [5]:

- Microprocessador de alto desempenho 1GHz ARM Cortex-A8 DaVinci DM3730 com acelerador de imagem e vídeo 720p HD DSP e acelerador gráfico PowerVR SGX com suporte para Open GL ES 2.0 e OpenVG;
- Porta para cartão microSD;
- Dispositivo Texas Instrument TPS65950 para gerenciamento de energia;
- Memória Flash de 1Gb do tipo *Package-on-package*.

As especificações completas do computador, da placa de extensão e da câmera podem ser encontrados nos links abaixo.

Overo Waterstorm COM:	https://store.gumstix.com/overo-waterstorm-com.html
Placa de extensão TOBI:	https://store.gumstix.com/tobi.html
Câmera Caspa VL:	https://store.gumstix.com/caspa-vl.html

2.4.2 Sistema operacional (SO)

O primeiro passo para a utilização desse computador, é a configuração e criação de uma imagem de sistema operacional que atende aos requisitos do projeto. São eles: compatibilidade com o computador utilizado, Overo COM, e suporte para aplicações em tempo real.

A técnica convencional para a criação de imagens desse tipo é a técnica de compilação cruzada (*cross-compiling*). Essa técnica consiste basicamente de compilar aplicações em uma máquina com

arquitetura diferente da máquina onde essa aplicação será executada. Nesse caso, a imagem do SO criada para arquitetura ARM é gerada em máquinas de arquitetura AMD64. Além do compilador, o ambiente de compilação cruzada é formado por diversas ferramentas, que servem para manipular código objeto em diferentes formatos.

Existem duas abordagens para a execução de aplicações de tempo real em Linux, uso de ferramentas que implementam um kernel duplo, como Xenomai ou RTAI (*Real Time Application Interface*), e uso de RTL (*Real-time Linux*). O uso de RTL vem crescendo muito nos últimos anos, assim como sua comunidade de desenvolvedores. Porém, ainda é uma ferramenta em evolução e ainda não apresenta desempenho tão bom como as ferramentas de kernel duplo [6]. Assim, escolheu-se utilizar a ferramenta Xenomai para esse projeto.

2.4.2.1 Xenomai

Atualmente, existe uma versão mais recentes do Xenomai que utiliza as capacidades em tempo real do kernel nativo do Linux, assim como RTL, dispensando assim o uso de um kernel duplo. Essa versão forma o núcleo *mercury* dentro do kernel do Linux. Já a versão que implementa o kernel duplo forma o núcleo *cobalt*. Nesse projeto, escolheu-se utilizar a versão *cobalt* [7].

O núcleo *cobalt* complementa o Linux com um co-kernel em tempo real, funcionando lado a lado com ele. Ele está integrado no kernel do Linux, lidando com todas as atividades críticas para o tempo, como lidar com interrupções e agendamento de threads em tempo real. O núcleo *cobalt* tem maior prioridade sobre as atividades nativas do kernel. Nesta configuração de kernel duplo, todas as aplicações de tempo real possuem interface com o núcleo Cobalt. Uma representação dessa configuração pode ser visualizada na figura [7].

A documentação completa do Xenomai pode ser encontrada em `https://xenomai.org/`.

2.4.2.2 Ferramenta RT-Mag

2.4.2.3 Geração de uma imagem de SO por compilação cruzada

2.4.3 Conexão serial

2.5 Comunicação com o piloto automático - Mavlink

2.5.1 Introdução

2.6 Rádios de comunicação - ???

2.6.1 Introdução

Capítulo 3

Resultados

Resumo opcional.

3.1 Introdução

Introduzir.

Capítulo 4

Conclusões

Concluir

4.1 Perspectivas Futuras

Perspectivas futuras

Referências Bibliográficas

[1] Hobby King. *goDiscover - User Manual*. USA: [s.n.], 2014.

[2] X-UAV Aeromodeling CO., LTD. *X-UAV - User Manual*. USA: [s.n.], 2015.

[3] PixHawk Community. *ArduPilot Autopilot Suite - Hardware, Firmware, Software & Community*. `http://ardupilot.org/ardupilot/index.html#`. [Online; acessado em Junho/2017].

[4] PX4 Development Community. *PX4 Development Guide*. `https://dev.px4.io/en/`. [Online; acessado em Junho/2017].

[5] Gumstix, Inc. *Gumstix Developer Center*. `https://gumstix.org/`. [Online; acessado em Junho/2017].

[6] BROWN, J. H. How fast is fast enough? choosing between xenomai and linux for real-time applications. 2016.

[7] Xenomai Development Community. *Xenomai Documentation*. `https://xenomai.org/start-here/#How_does_Xenomai_deliver_real-time`. [Online; acessado em Junho/2017].

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

Descrever CD.

II. PROGRAMAS UTILIZADOS

Quais programas foram utilizados?