



TRABALHO DE GRADUAÇÃO 1

DESENVOLVIMENTO DE UM SISTEMA COM VEÍCULOS AÉREOS NÃO-TRIPULADOS AUTÔNOMOS

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

**DESENVOLVIMENTO DE UM SISTEMA COM
VEÍCULOS AÉREOS NÃO-TRIPULADOS AUTÔNOMOS**

Eduardo Moura Cirilo Rocha

Orientador: Prof. Henrique Cezar Ferreira, ENE/UNB

Brasília, Junho de 2017

SUMÁRIO

1 INTRODUÇÃO	1
1.1 CONTEXTUALIZAÇÃO.....	1
1.2 OBJETIVOS DO TRABALHO	2
2 DESCRIÇÃO DO SISTEMA	3
2.1 INTRODUÇÃO	3
2.2 AERONAVES.....	3
2.3 PILOTO AUTOMÁTICO - PIXHAWK	4
2.3.1 INTRODUÇÃO.....	4
2.3.2 INSTALAÇÃO DE FIRMWARE.....	6
2.3.3 CALIBRAÇÃO DE SENsoRES	7
2.3.4 DIAGRAMAS DE CONEXões	7
2.3.5 POUSO AUTOMÁTICO (SENSOR LIDAR)	9
2.4 COMPUTADOR EMBARCADO - GUMSTIX OVERO WATERSTORM COM.....	10
2.4.1 INTRODUÇÃO.....	10
2.4.2 SISTEMA OPERACIONAL (SO).....	11
2.4.3 XENOMAI	12
2.4.4 PROJETO RT-MAG	12
2.5 COMUNICAÇÃO COM O PILOTO AUTOMÁTICO - MAVLINK	13
2.5.1 INTRODUÇÃO.....	13
2.5.2 MENSAGENS MAVLINK.....	13
2.5.3 FUNCIONAMENTO DO MAVLINK	13
2.5.4 CONEXÃO FÍSICA COM O PILOTO AUTOMÁTICO	14
2.6 RÁDIOS DE COMUNICAÇÃO - PICO SERIES P900 ENCAPSULADO	14
2.6.1 INTRODUÇÃO.....	14
2.6.2 ESPECIFICAÇÕES DO P900.....	15
2.6.3 CONFIGURAÇÃO DOS RÁDIOS	15
2.7 DIAGRAMA COMPLETO DE UM VANT.....	18
3 RESULTADOS	19
3.1 INTRODUÇÃO	19
3.2 TESTE DE VOO DAS AERONAVES.....	19
3.3 TESTE DE CONEXÃO MAVLINK	19

3.4 TESTE DE FUNCIONAMENTO DOS RÁDIOS DE COMUNICAÇÃO P900.....	21
3.5 TESTE DE FUNCIONAMENTO DO PILOTO AUTOMÁTICO PIXHAWK	21
3.6 GUMSTIX OVERO COM.....	23
4 CONCLUSÕES	25
4.1 TRABALHOS FUTUROS	26

LISTA DE FIGURAS

2.1	Foto do avião <i>X-UAV Skua FPV Plane</i> [1].....	4
2.2	Foto do avião <i>Go Discover FPV Plane</i> [2].....	4
2.3	Piloto automático Pixhawk [3].	5
2.4	Programa de Estação de Controle Terrestre QGroundControl.	6
2.5	Exemplo de calibração de sensores pelo programa QGroundControl.	8
2.6	Esquema de montagem do sensor lidar via conexão PWM [5].	9
2.7	Diagrama de conexão do Pixhawk.....	9
2.8	Sistema Gumstix com computador Overo WaterSTORM, placa de expansão Tobi e câmera Caspa VL.....	11
2.9	Representação de funcionamento do núcleo <i>cobalt</i> do Xenomai [11].	12
2.10	<i>Modem</i> Pico Series P900 [15].....	15
2.11	Visualização do programa HyperTerminal ao se conectar o <i>modem</i> em modo de comando [15].	16
2.12	Visualização do programa HyperTerminal durante a configuração de um <i>modem</i> [15].	17
2.13	Diagrama completo dos componentes de um VANT projetado.	18
3.1	Foto do avião <i>X-UAV Skua FPV Plane</i> na pista de decolagem.....	22
3.2	Foto do avião <i>X-UAV Skua FPV Plane</i> durante teste de voo.	22
3.3	ECT QGroundControl conectada ao avião por meio dos rádios de telemetria na pista de voo.....	23

Capítulo 1

Introdução

1.1 Contextualização

Esse trabalho faz parte de um projeto de pesquisa que visa o estudo e aplicação de técnicas de controle cooperativo para o controle de sistemas com aeronaves autônomas. Esse projeto faz parte do Laboratório de Robótica Aérea da Universidade de Brasília.

Nesse projeto, utiliza-se aeronaves de pequeno porte que podem ser chamadas de veículos aéreos não tripulados (VANTs). O uso de VANTs cresceu muito nos últimos anos devido a uma grande queda no preço dos componentes desses sistemas, como sensores e computadores embarcados. Antigamente, o alto custo restringia o seu uso para aplicações militares. Atualmente, o uso de VANTs é muito amplo, utiliza-se VANTs também para aplicações de segurança, aplicações agrícolas, operações de mapeamento, etc.

Um VANT autônomo possui equipamentos embarcados capazes de realizar o controle dele mesmo, como, por exemplo, um piloto automático e um computador embarcado. Por meio de rádios de comunicação, VANTs são capazes de se comunicar com estações de controle ou até mesmo entre eles mesmos. Essas estações de controle podem enviar comandos de voo para as aeronaves e receber informações do voo, como dados de telemetria.

Esse projeto utiliza pilotos automáticos para o controle dos VANTs. Esses dispositivos são capazes de ler os dados provindos dos sensores e estimar o estado atual da aeronave. Pela análise desse estado atual da aeronave e dos comandos de voo recebidos, o piloto automático é capaz de determinar qual devem ser os estados dos atuadores do sistema, como posições das superfícies de controle e velocidade do motor. Esses dispositivos são capazes de implementar diversos modos de voo, de modos de voo que apenas auxiliam um piloto à modos de voo totalmente autônomos.

1.2 Objetivos do trabalho

Esse trabalho tem como objetivo a construção de um sistema autônomo com três aeronaves e uma estação terrestre. Recentemente, comprou-se três aeronaves e todos os componentes necessários para a construção desse sistema.

Primeiramente, deve-se aprender a utilizar todos os componentes que fazem partes desse sistema, como pilotos automáticos, *modens* de comunicação, sensores das aeronaves, etc. Após o correto funcionamento de todos esses componentes, deve-se montar todas as aeronaves e verificar o correto funcionamento do sistema, inclusive a comunicação entre as aeronaves.

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 como ocorre a comunicação entre eles.

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 2.1.

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 2.2.



Figura 2.1: Foto do avião *X-UAV Skua FPV Plane* [1].

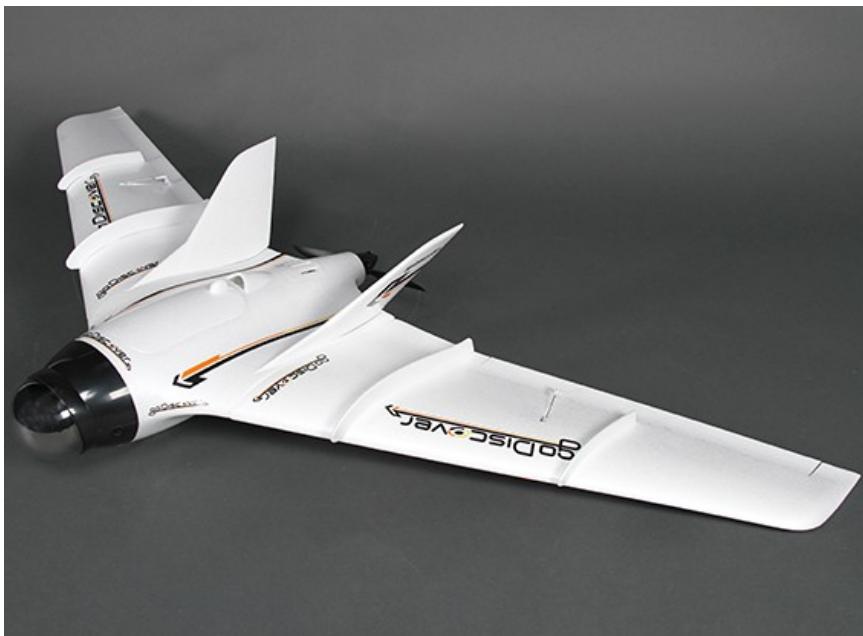


Figura 2.2: Foto do avião *Go Discover FPV Plane* [2].

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 e 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. O piloto automático Pixhawk pode ser visto na figura 2.3. [3]



Figura 2.3: Piloto automático Pixhawk [3].

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.

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 encontradas em [3].

2.3.2 Instalação de firmware

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 por meio da comunicação com 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. O programa QGroundControl pode ser visto na figura 2.4.

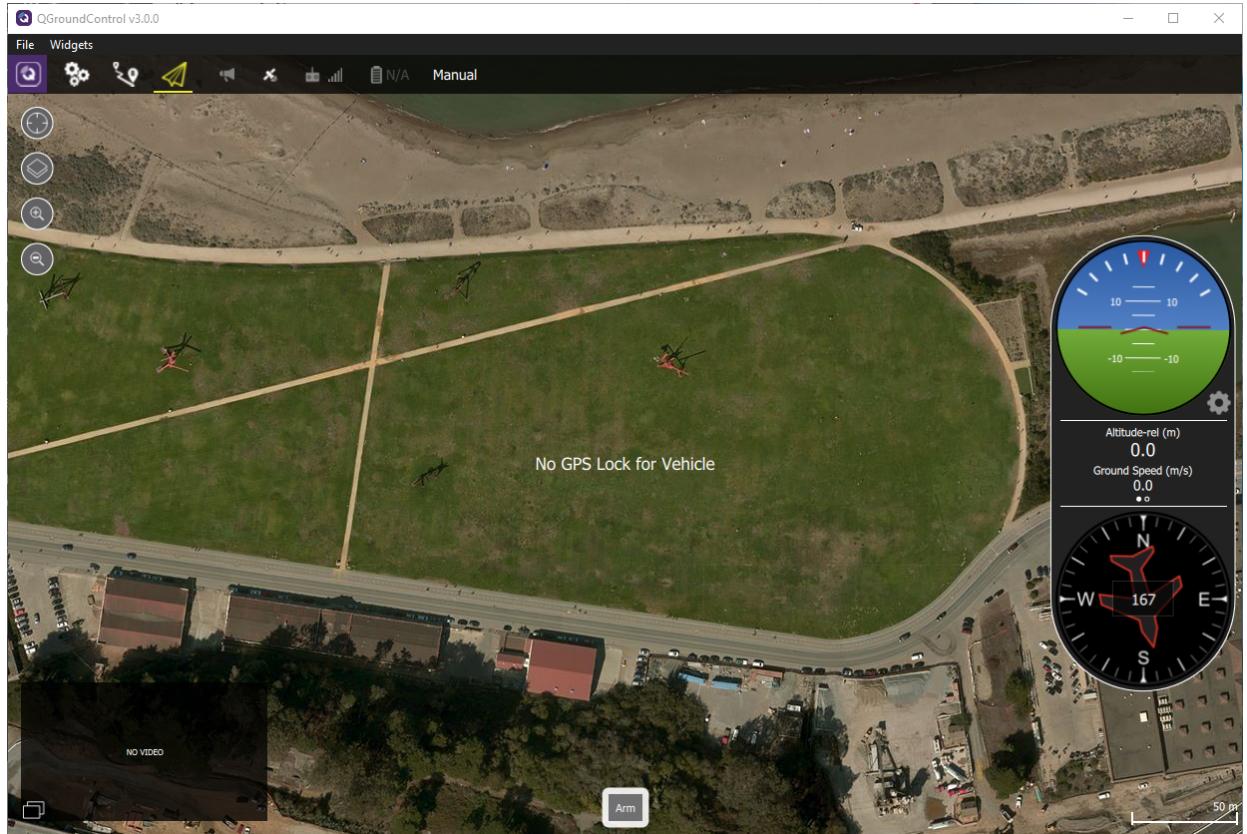


Figura 2.4: Programa de Estação de Controle Terrestre QGroundControl.

O 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].

O desenvolvimento de aplicações ou alteração do *firmware* padrão pode ser feito em qualquer sistema operacional, mas recomenda-se o uso de Linux ou Mac OS X. A documentação completa sobre desenvolvimento de aplicações pode ser encontrado em [4].

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 esses 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. Um exemplo de calibração de sensores pelo programa QGroundControl pode ser visto na figura 2.5.

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 tipo de projeto. A redundância garante também que, quando um dos sensores para de funcionar, existe outro sensor no sistema capaz de substituir ele. Isso previne a queda de aeronaves por falhas de sensores. Em aviões comerciais da companhia Embraer, por exemplo, há uma redundância de pelo menos seis sensores de cada tipo. [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

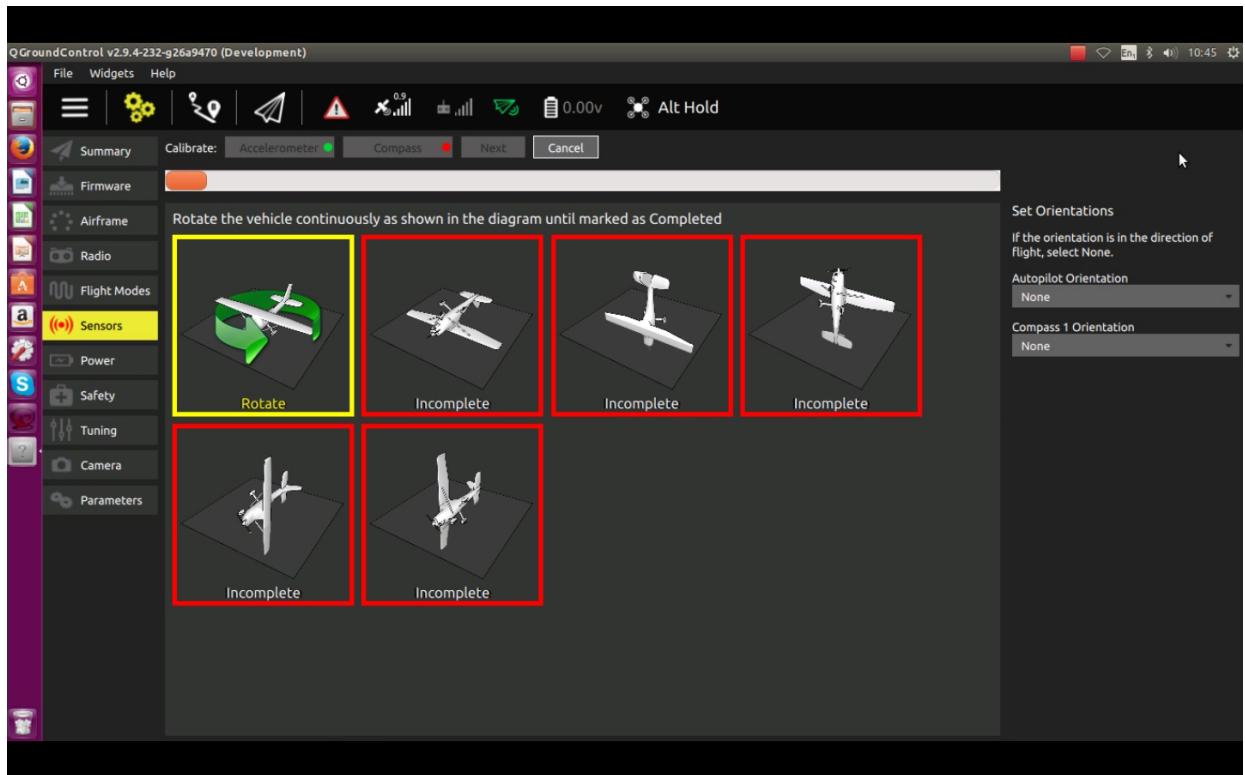


Figura 2.5: Exemplo de calibração de sensores pelo programa QGroundControl.

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 visto na tabela 2.1 e o esquema de montagem pode ser visto na figura 2.6, onde o valor do resistor pode variar entre 200Ω e $1k\Omega$ [3]. Mais detalhes sobre a conexão podem ser encontrados em [5].

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 [6], como, por exemplo, energização triplo-redundante. [6]

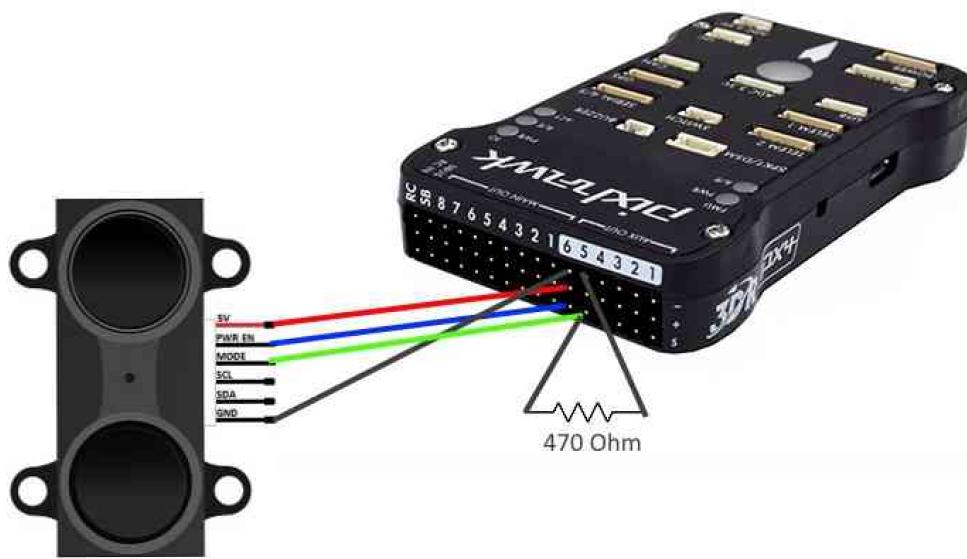


Figura 2.6: Esquema de montagem do sensor lidar via conexão PWM [5].

Um diagrama geral de dispositivos conectados ao Pixhawk pode ser visto na figura 2.7. Os servo motores, o *electronic speed controller* (ESC), o lidar e o receptor de rádio devem ser conectados a trilha PWM do Pixhawk. Os demais componentes devem ser conectados as portas que contém seu nome.

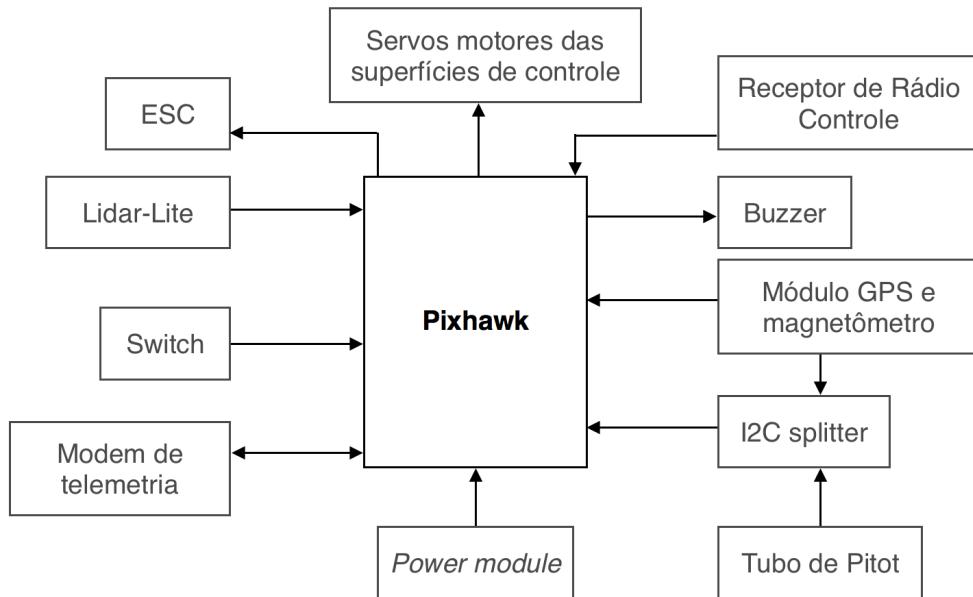


Figura 2.7: Diagrama de conexão do Pixhawk.

2.3.5 Pouso automático (sensor Lidar)

Após a conexão do lidar ao sistema via PWM, alguns parâmetros do piloto automático devem ser alterados para que ele reconheça o sensor. Esses parâmetros podem ser facilmente alterados

por um programa de ECT. São eles: [5]

- RNG_FND = 5, indica que a conexão ocorre via PWM.
- RNDFND_MAX_CM = 4000, representa a distância máxima em que o sensor é confiável.
- RNDFND_STOP_PIN = 55, pino conectado ao sinal de ativação do lidar. Permite que o dispositivo reinicie o sensor caso ele pare de fornecer dados.
- Os parâmetros RNDFND_SCALING e RNDFND_OFFSET devem ser ajustados de forma a se calibrar o sensor (costumam ser aproximadamente 0 e 1, respectivamente).

O sensor pode ser testado pela ECT, onde as leituras podem ser observadas no campo *sonar-range*. Após a configuração do sensor, o piloto automático é capaz de pousar o avião de forma muito mais rápida e precisa. O pouso ocorre pelo envio do comando *Land* ao controlador, mas para que ele ocorra corretamente deve-se definir a posição da pista de pouso e deve-se ajustar os parâmetros de pouso, como por exemplo a velocidade com que o avião deve pousar. Durante esse trabalho não foi realizado nenhum pouso automático. A documentação detalhada sobre poucos automáticos pode ser encontrada em [7].

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 2.8, 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.

Algumas das especificações mais importantes do Overo Waterstorm COM são [8]:

- 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;



Figura 2.8: Sistema Gumstix com computador Overo WaterSTORM, placa de expansão Tobi e câmera Caspa VL.

- 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 [9]. Assim, escolheu-se utilizar a ferramenta Xenomai para esse projeto. [10]

2.4.3 Xenomai

Atualmente, existe uma versão mais recente 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*. [11]

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 2.9. [11]

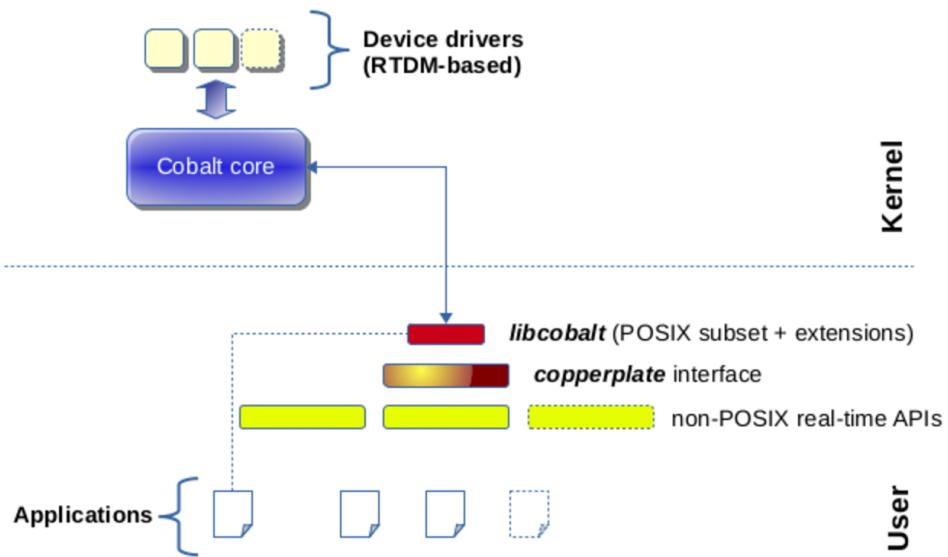


Figura 2.9: Representação de funcionamento do núcleo *cobalt* do Xenomai [11].

A documentação completa do Xenomai pode ser encontrada em [11].

2.4.4 Projeto RT-MaG

O projeto RT-MaG (*Real-Time - Marseille Grenoble Project*) é um projeto desenvolvido pelo Gipsa-Lab (Grenoble, França) e o Institute of Mouvement Sciences (ISM, Marseille, França). O objetivo deste projeto é fornecer ferramentas eficientes para a prototipagem rápida de robôs para pesquisa e aplicações acadêmicas. O RT-MaG fornece uma caixa de ferramentas para Matlab e

Simulink para programar sistemas Linux-COM. Com a ferramenta, pode-se facilmente gerar um aplicativo autônomo em tempo real a partir de um modelo Simulink para um robô usando um sistema Linux.

Essas ferramentas consistem em um conjunto de blocos simulink que fornecem acesso direto às entradas e saídas de uma COM. Atualmente, a Gumstix Overo COM é totalmente suportada. Os modelos Simulink são convertidos automaticamente em aplicações em tempo real. O uso dessas ferramentas é totalmente gratuito. [12]

Pode-se encontrar imagens de SO já compiladas para a Gumstix Overo COM em [12], onde encontram-se imagens geradas de distribuições Linux provindas do Projeto Yocto.

2.5 Comunicação com o piloto automático - MAVLink

2.5.1 Introdução

MAVLink (*Micro Air Vehicle Communication Protocol*) é um protocolo de comunicação leve feito para o uso em veículos aéreos pequenos. Ele capaz de enviar estruturas de dados em canais seriais com alta eficiência e enviar esses pacotes para a estação de controle de terra ou computador embarcado. Ele é amplamente usado na plataforma Pixhawk, o piloto automático escolhido para esse projeto. Apesar de seu nome, o uso deste protocolo tem se expandido muito nos últimos anos e ele é usado também para a comunicação com robôs terrestres. [13]

2.5.2 Mensagens MAVLink

Uma mensagem MAVLink é basicamente um conjunto de bytes codificados pela ECT e enviado para o piloto automático, ou vice-versa, via conexão USB serial ou telemetria, mas não pelos dois ao mesmo tempo (quando os dois estão conectados, a conexão ocorrerá via porta USB e a telemetria será ignorada). [13]

Cada pacote MAVLink possui 17 bytes de tamanho, onde 6 correspondem ao cabeçalho, 9 correspondem aos dados a serem transmitidos, chamados de *payload*, e 2 correspondem a bytes de *checksum*. A tabela 2.2 detalha as informações contidas no cabeçalho de uma mensagem MAVLink. Os bytes de *checksum* são utilizados para a detecção de erros durante a transmissão das mensagens. [14]

2.5.3 Funcionamento do MAVLink

MAVLink pode ser definido como nada mais do que a estrutura das mensagens enviadas, uma sequência de bytes. Essas mensagens são recebidas pelo piloto automático ou pela ECT pela interface de *hardware* (USB serial ou telemetria) e são decodificadas em *software*. [13]

O que realmente interessa nessas mensagens é o *payload* junto com a identificação da mensagem, que indica o significado desse *payload*. Para isso, há três passos para se interpretar uma mensagem

byte 0	Início da mensagem, sempre 0xFE.
byte 1	Tamanho da mensagem, igual a 9.
byte 2	Número da sequencia, indica a sequência das mensagens enviadas.
byte 3	Identificação do sistema (<i>System_ID</i>), indica qual sistema está enviando a mensagem.
byte 4	Identificação do componente (<i>Component_ID</i>), indica qual componente do sistema está enviando a mensagem.
byte 5	Identificação da mensagem (<i>Message_ID</i>), indica qual o conteúdo da mensagem.

Tabela 2.2: Descrição do cabeçalho de uma mensagem MAVLink [14].

desse tipo [13]:

1. Primeiramente, a mensagem passa por um método de segurança, chamado de *handlemessage* na documentação oficial, que lê a identificação do sistema e do componente.
2. Em seguida, o *payload* é extraído e colocado dentro de um pacote.
3. Finalmente, lê-se a identificação da mensagem e coloca-se esse pacote em uma estrutura de dados apropriada para aquele tipo de mensagem. Essas estruturas de dados devem ser perfeitamente iguais nos dispositivos emissor e receptor.

Um exemplo de programa que implementa uma comunicação com MAVLink pode ser encontrado no repositório em https://github.com/mavlink/c_uart_interface_example. Esse programa cria uma comunicação serial com o Pixhawk e troca mensagens com ele. A implementação da comunicação entre o Pixhawk e o computador embarcado nesse projeto é baseada nesse exemplo. Dentro do repositório tem um arquivo de texto que explica como compilar e rodar o programa.

2.5.4 Conexão física com o piloto automático

A conexão pode ser realizada de duas formas, pela entrada mini-USB do Pixhawk ou pela porta de comunicação serial por meio de um cabo FTDI. Recomenda-se a comunicação por meio da porta serial. O uso da entrada mini-USB do pixhawk desativa a porta de telemetria do dispositivo, a não ser que isso seja alterado em seu *firmware*.

2.6 Rádios de comunicação - Pico Series P900 Encapsulado

Esta seção foi inteiramente baseada na referência [15], a documentação dos módulos P900.

2.6.1 Introdução

Um *modem* P900 é capaz de fornecer comunicação em série sem fio de alto desempenho em topologias robustas de ponto a ponto, em malha ou ponto a multiponto. Esses rádios são capazes de realizar comunicação de longo alcance e podem atingir velocidades de transmissão muito rápidas.

Módulos dessa linha operam dentro da banda de frequências 902-928 MHz usando tecnologia FHSS, provendo comunicação assíncrona e sem fio entre equipamentos com interface de comunicação serial.

Por serem pequenos e com excelente performance, esses rádios são ideais para esse projeto, pois eles devem estar embarcados em todas as aeronaves.

O módulo utilizado pode ser visto na figura 2.10, onde à esquerda encontra-se a versão utilizada, que é encapsulada.



Figura 2.10: *Modem Pico Series P900* [15].

2.6.2 Especificações do P900

É importante mencionar que esse modelo é compatível com nível lógico de 3.3V, que corresponde com o nível lógico da porta USB do computador Gumstix Overo COM. Assim, esses *modem* pode ser conectado diretamente no computador, sem a necessidade de conversores de nível lógico.

A tabela 2.3 mostra algumas das especificações mais importantes a serem levadas em consideração:

Frequência de operação	902 - 928 MHz
Detecção de erro	32 bits de CRC, ARQ
Alcance	40 milhas (\approx 64 km)
Potência de saída	de 100mW a 1W
Velocidade de comunicação	até 230,4 kbps assíncrono
Temperatura de operação máxima	até 85°C
Voltagem de entrada (Vin)	8 - 30Vdc

Tabela 2.3: Especificações dos módulos Pico Series P900 [15].

2.6.3 Configuração dos rádios

Qualquer módulo P900 pode ser configurado como coordenador primário, coordenador secundário, coordenador ocioso ou remoto (escravo) em topologias de ponto a ponto, em malha ou ponto a multiponto. Como todas as aeronaves precisam se comunicar entre si, optou-se por utilizar a topologia em malha. Assim, o emissor pode enviar um pacote de dados para todos os receptores

ao mesmo tempo. Por ser uma rede pequena, decidiu-se também definir apenas um coordenador primário e 3 módulos remotos. Uma explicação detalhada sobre todas as topologias de operação e tipos de configurações dos rádios pode ser encontrada documentação do rádio, referência [15].

O módulo possui LEDs que auxiliam na determinação de seu estado atual. A tabela 2.4 descreve o significado de cada LED.

PWR (azul)	Aceso quando o módulo está conectado a uma fonte de energia.
485 (azul)	Aceso quando a porta de dados do módulo está configurada como uma porta RS485.
TX LED (vermelho)	Aceso quando o <i>modem</i> está transmitindo dados.
RX LED (verde)	Quando aceso, significa que o módulo está sincronizado e recebeu pacotes válidos.
RSSI (3x Green)	O número de LEDs acesos indica a força do sinal.

Tabela 2.4: Significado dos LEDs no módulo P900 encapsulado [15].

Para se configurar os *modems*, eles devem estar em modo de comando e conectados a um computador via conexão serial. Para isso, utilizou-se o programa para Windows HyperTerminal. Há dois modos de colocar um módulo em modo de comando:

1. Conectar o módulo desligado (desconectado de sua fonte de alimentação) em um computador via conexão serial configurada para 9600 bps. Em seguida, ligar o módulo enquanto seu botão CONFIG está sendo pressionado.
2. Conectar o módulo ligado em um computador via conexão serial configurada para 9600 bps. Em seguida, digitar '+++’ e esperar um segundo.

Após entrar no modo de comando, o monitor vai mostrar a resposta do módulo 'NO CARRIER OK', como na figura 2.11. Para sair do modo de comando, basta digitar 'ATA' e pressionar a tecla 'Enter'.

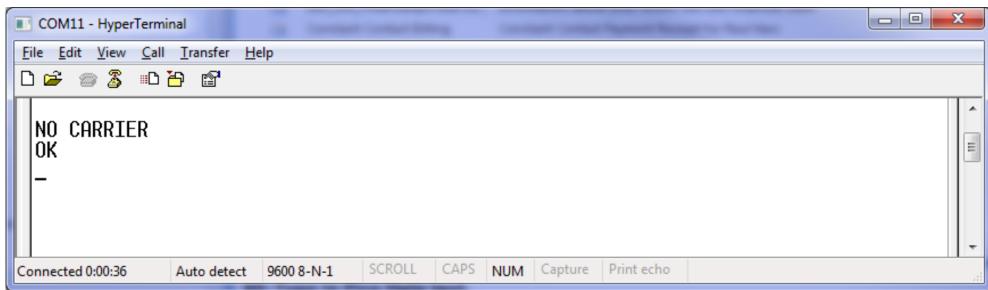


Figura 2.11: Visualização do programa HyperTerminal ao se conectar o *modem* em modo de comando [15].

Em modo comando, pode-se verificar todas as configurações de um módulo pelos valores de seus registradores. Por exemplo, o registrador S101 indica qual o modo de operação desse módulo, 2 para remoto e 4 para coordenador primário. Para navegar entre as configurações dos módulos e

altera-las, deve-se utilizar os comandos de fábrica. A figura 2.12 mostra exemplos da utilização de comandos, onde as marcações na figura significam:

- A) Comando AT&F1 - Definir registradores para padrão de fábrica de coordenador primário.
- B) Comando AT&W - Escrever mudanças na memória.
- C) Comando AT&V - Mostrar configurações (imprimi na tela os dados que estão na parte branca da imagem).
- D) Registrador S101 - Modo de operação está definido como 4, modo de coordenador primário.
- E) Registrador S104 - Endereço de rede em seu valor padrão de fábrica, todas as unidades de rede da malha devem ter esse mesmo valor. Recomenda-se mudar para um valor diferente do padrão.
- F) Registrador S133 - Tipo de rede, 2 ou 3 para em malha.
- G) Registrador S140 - Endereço de destino. O padrão é FF:FF:FF:FF:FF:FF que significa que os pacotes serão enviados para todos os dispositivos na rede. Isso pode ser alterado para o endereço MAC de um dispositivo específico.
- H) MAC - Endereço MAC desse dispositivo definido em fábrica.

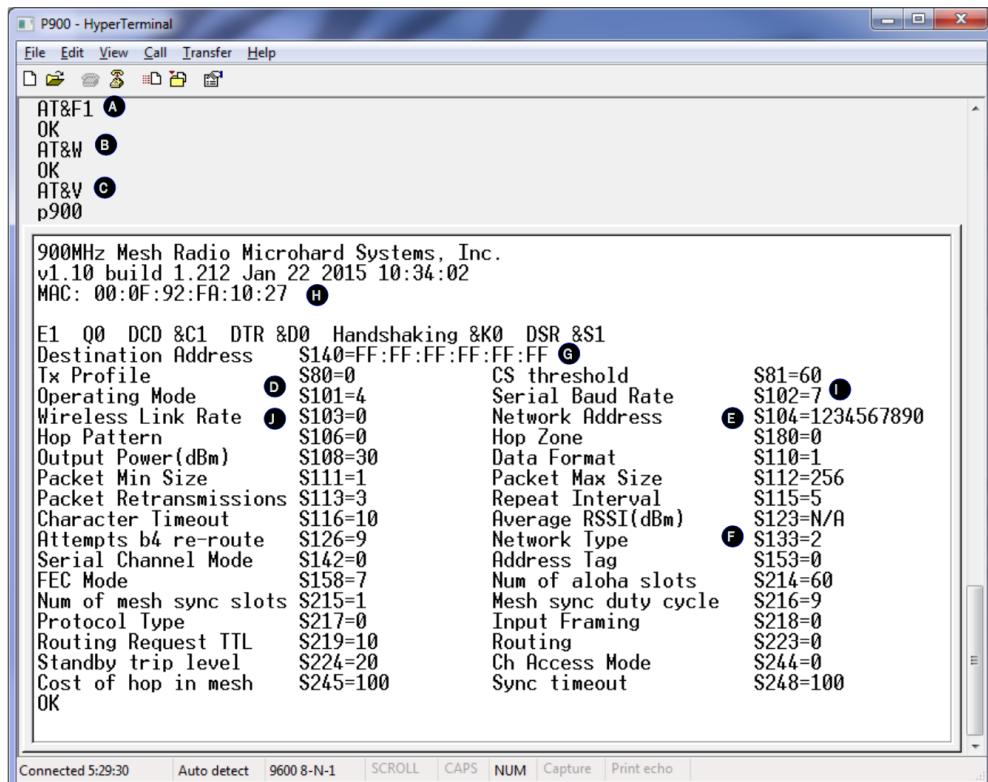


Figura 2.12: Visualização do programa HyperTerminal durante a configuração de um *modem* [15].

Mais exemplos de configurações podem ser encontradas na documentação, referência [15], assim como configurações avançadas de como evitar colisões na rede, de roteamento de rede e de sincronização de rede. Como a rede em questão é pequena e simples, decidiu-se não detalhar essas configurações aqui, pois elas não serão utilizadas no estado atual do projeto.

2.7 Diagrama completo de um VANT

Um diagrama completo de um VANT autônomo projetado pode ser visto na figura 2.13. Para simplificação do diagrama, representou-se o Pixhawk e alguns elementos ligados a ele como um só bloco. Um diagrama mais detalhado das conexões desses componentes ao Pixhawk pode ser visto na figura 2.7. Os servo motores, o *electronic speed controller* (ESC), o lidar e o receptor de rádio devem ser conectados a trilha PWM do Pixhawk. Os demais componentes conectados ao Pixhawk devem ser conectados as portas que contém seu nome. A conexão entre o Pixhawk e o computador Gumstix Overo COM deve ocorrer por um cabo USB ou FTDI, conforme mencionado na seção 2.5. O *modem* P900 pode ser conectado à Gumstix Overo COM por um cabo serial RS232-USB.

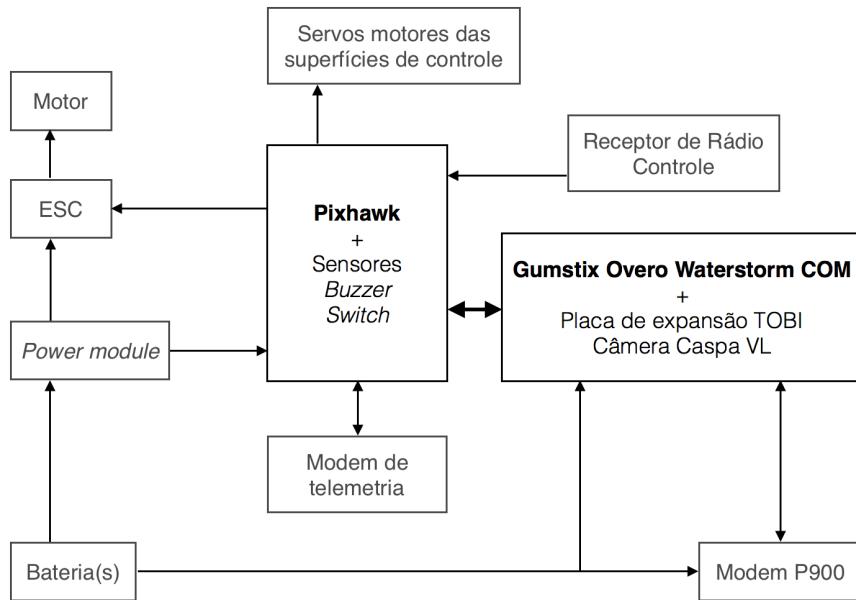


Figura 2.13: Diagrama completo dos componentes de um VANT projetado.

Capítulo 3

Resultados

3.1 Introdução

Com excessão do computador embarcado, todos os outros componentes do sistema foram testados e funcionaram corretamente. Houve problemas com a criação da imagem de SO para a Gumstix Overo COM. Assim, não foi possível realizar um voo do sistema completo.

As seções abaixo descrevem os testes realizados as componentes do sistema e os resultados obtidos.

3.2 Teste de voo das aeronaves

Realizou-se testes de voo de todas as aeronaves. A comunicação com a aeronave ocorreu diretamente pelo rádio controle e pelo receptor de rádio, que foi diretamente conectado as servos motores das superfícies de controle da aeronave e ao ESC do motor.

Todos os voos foram realizados com sucesso. As aeronaves se demonstraram estáveis e fáceis de se pilotar.

3.3 Teste de conexão MAVLink

Testou-se a conexão de um computador com sistema operacional Linux, equivalente a Gums-tix Overo COM, com um Pixhawk por meio do protocolo MAVLink. Esse teste foi feito por meio de uma adaptação ao exemplo encontrado no link: https://github.com/mavlink/c_uart_interface_example. O código consiste dos seguintes passos:

1. Abrir conexão serial com a porta USB conectada ao Pixhawk;
2. Checar por mensagem do tipo HEARTBEAT do Pixhawk;
3. Pegar parâmetros SYSTEM_ID e COMPONENT_ID do Pixhawk;

4. Pegar Posição do Pixhawk oito vezes a uma taxa de uma vez por segundo;
5. Encerrar conexão.

O resultado do teste encontra-se no arquivo `resultado_MAVLink.txt`, que pode ser visto abaixo.

`resultado_MAVLink.txt`

```
OPEN PORT
Connected to /dev/ttyUSB0 with 57600 baud, 8 data bits, no parity, 1 stop bit 8N1

START READ THREAD

CHECK FOR HEARTBEAT
Found

GOT VEHICLE SYSTEM ID: 1
GOT AUTOPILOT COMPONENT ID: 50

INITIAL POSITION XYZ = [ 0.0035 , 0.0901, -0.9980 ]
INITIAL POSITION YAW = 1.0943

START WRITE THREAD

ENABLE OFFBOARD MODE

SEND OFFBOARD COMMANDS
POSITION SETPOINT XYZ = [ -4.9965 , -4.9099 , -0.9980 ]
POSITION SETPOINT YAW = 1.0943
0 CURRENT POSITION XYZ = [ 0.0035 , 0.0901, -0.9980 ]
1 CURRENT POSITION XYZ = [ 0.0030 , 0.0899, -0.9990 ]
2 CURRENT POSITION XYZ = [ 0.0024 , 0.0901, -0.9987 ]
3 CURRENT POSITION XYZ = [ 0.0024 , 0.0911, -0.9986 ]
4 CURRENT POSITION XYZ = [ 0.0024 , 0.0909, -0.9979 ]
5 CURRENT POSITION XYZ = [ 0.0033 , 0.0910, -0.9983 ]
6 CURRENT POSITION XYZ = [ 0.0037 , 0.0900, -0.9991 ]
7 CURRENT POSITION XYZ = [ 0.0033 , 0.0899, -0.9988 ]

DISABLE OFFBOARD MODE

CLOSE THREADS

CLOSE PORT
```

3.4 Teste de funcionamento dos rádios de comunicação P900

O teste funcionamento dos rádios ocorreu pela comunicação entre computadores com sistema operacional Windows por meio do programa HyperTerminal. O mesmo deve funcionar corretamente em computadores com sistema operacional Linux pela abertura de conexões seriais entre os computadores e os *modems*. Configurou-se um dos rádios como mestre primário e os outros três como remotos (escravos).

Durante o teste, um dos rádio transmitiu mensagens para os outros três, que receberam as mensagens corretamente. Transmitiu-se mensagens de 8 bits, correspondentes a caracteres. Os caracteres recebidos pelos rádios encontram-se nos arquivos *recebido_radio1.txt*, *recebido_radio2.txt* e *recebido_radio3.txt*, que podem ser vistos abaixo.

recebido_radio1.txt

TESTE DE CONEXAO ENTRE OS MODENS.
ESSES CARACTERES ESTAO SENDO TRANSMITIDOS ENTRE OS RADIOS,
CARACTERE POR CARACTERE.

recebido_radio2.txt

TESTE DE CONEXAO ENTRE OS MODENS.
ESSES CARACTERES ESTAO SENDO TRANSMITIDOS ENTRE OS RADIOS,
CARACTERE POR CARACTERE.

recebido_radio3.txt

TESTE DE CONEXAO ENTRE OS MODENS.
ESSES CARACTERES ESTAO SENDO TRANSMITIDOS ENTRE OS RADIOS,
CARACTERE POR CARACTERE.

3.5 Teste de funcionamento do piloto automático Pixhawk

Inicialmente, conectou-se o Pixhawk ao programa de ECT QGroundControl. Instalou-se o *firmware* apropriado para a aeronave em questão, calibrou-se todos os sensores e configurou-se e calibrou-se o lidar. Realizou-se um voo com um avião do modelo *X-UAV Skua FPV Plane* e salvou-se toda a telemetria de voo. Por erro de fixação do Pixhawk à aeronave, o piloto automático soltou-se durante o voo. Por isso, os dados de telemetria salvos ficaram ruins e não apresentam utilidade. Pode-se ver na figura 3.3 a ECT conectada ao avião por meio dos rádios de telemetria antes do voo realizado. Fotos do voo podem ser vistas nas figuras 3.1 e 3.2.



Figura 3.1: Foto do avião *X-UAV Skua FPV Plane* na pista de decolagem.



Figura 3.2: Foto do avião *X-UAV Skua FPV Plane* durante teste de voo.

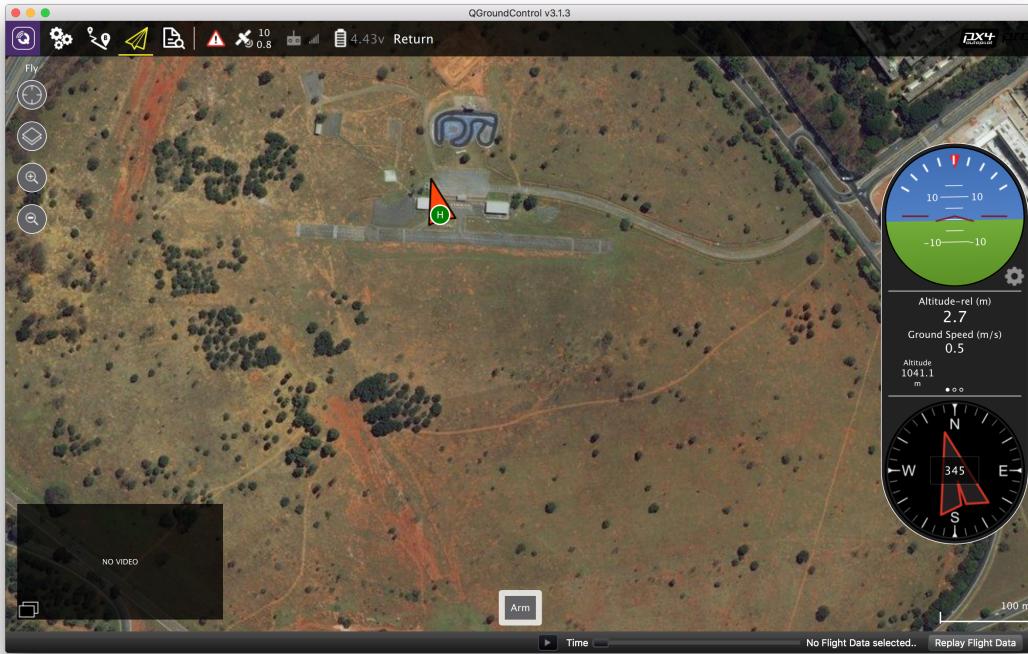


Figura 3.3: ECT QGroundControl conectada ao avião por meio dos rádios de telemetria na pista de voo.

3.6 Gumstix Overo COM

Conseguiu-se gerar imagens de SO para a Gumstix Overo COM corretamente por meio das distribuições fornecidas pelo projeto Yocto. Gerou-se, com sucesso, imagens com a ferramenta RT-MaG e com suporte à câmera Caspa VL. Houve problemas com a instalação da ferramenta Xenomai. Seguiu-se todos os passos fornecidos pela documentação da ferramenta, mas a geração da imagem apresentou diversos problemas, muitos relacionados ao número das versões das ferramentas utilizadas durante o processo de compilação cruzada. Muitos desses erros foram resolvidos e consertados, mas não todos. Por isso, ainda não há uma imagem de SO funcional com a ferramenta Xenomai.

Capítulo 4

Conclusões

Durante esse trabalho, aprendeu-se a utilizar todos os componentes essenciais para que o sistema funcione. Conforme mencionado anteriormente, todos esses componentes foram testados e utilizados corretamente, com exceção do computador Gumstix Overo COM. Montou-se uma aeronave quase completa, mas sem o computador embarcado e sem os *modens* de comunicação.

A utilização do piloto automático Pixhawk é relativamente simples, pois sua documentação oficial é muito vasta e completa. Além de sua documentação oficial, pode-se encontrar muitos materiais e fóruns sobre o Pixhawk online. Já o computador Gumstix Overo COM não apresenta essa mesma simplicidade. Além de sua documentação oficial, que é antiga e não é muito completa, não se encontra muitos materiais relacionados na internet. Por exemplo, todos os materiais encontrados sobre a geração de imagens de SO são antigos, de 2011 para trás. Esse é o motivo de a geração de uma imagem com a ferramenta Xenomai gerar tantos erros relacionados ao número das versões das ferramentas utilizadas durante o processo de compilação cruzada, todos os materiais sobre isso consideram o uso de versões mais antigas dessas ferramentas.

Vale mencionar que o desenvolvimento com o Pixhawk foi atrasado pela falta de um piloto de aviões desse tipo dentro da equipe do laboratório. Apesar de que o Pixhawk estava pronto para ser testado há mais de meses antes da finalização desse trabalho, somente durante as duas últimas semanas foi possível a realização do primeiro voo teste. Isso impediu a realização de testes de voos autônomos individuais, testes de poucos autônomos com o uso do sensor lidar e obtenção de dados de voos por meio de telemetria para a determinação dos parâmetros dos modelos matemáticos das aeronaves.

4.1 Trabalhos futuros

Primeiramente, deve-se priorizar o computador embarcado Gumstix Overo COM, pois ele é o único componente que ainda não está funcionando. Assim, ele impede com que o sistema seja completamente montado. Deve-se gerar uma imagem de SO Linux com as ferramentas RT-Mag e Xenomai e com suporte para a câmera Caspa VL. Após a geração bem sucedida dessa imagem de SO, deve-se testar a conexão do computador com o Pixhawk via o protocolo de comunicação MAVLink. Como já foram realizados testes com o uso de outros computadores Linux com sucesso, esse teste deve ocorrer de forma rápida e sem problemas. Em seguida, deve-se testar a comunicação entre a Gumstix Overo COM com os *modens* de comunicação, que também deve ocorrer sem problemas, pois isso já foi testado com outros computadores.

Deve-se também realizar mais voos de teste para o Pixhawk, sem o computador embarcado. Deve-se realizar testes de voos autônomos e testes de poucos autônomos com o uso do sensor lidar. Esses voos também vão proporcionar prática com o uso das aeronaves e todos os procedimentos relacionados.

Finalmente, deve-se realizar voos de teste com o sistema completo, com um computador embarcado em cada aeronave, onde deve-se testar a comunicação por meio dos *modens* durante o voo.

Referências Bibliográficas

- [1] X-UAV Aeromodeling CO., LTD. *X-UAV - User Manual*. USA: [s.n.], 2015.
- [2] Hobby King. *goDiscover - User Manual*. USA: [s.n.], 2014.
- [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] PixHawk Community. *Pixhawk Documentation - LIDAR-Lite Rangefinder*. <http://ardupilot.org/copter/docs/common-rangefinder-lidarlite.html?highlight=lidar#>. [Online; acessado em Junho/2017].
- [6] PixHawk Community. *Pixhawk Documentation - Powering the Pixhawk*. <http://ardupilot.org/copter/docs/common-powering-the-Pixhawk.html#common-powering-the-Pixhawk>. [Online; acessado em Junho/2017].
- [7] PixHawk Community. *Pixhawk Documentation - Automatic Landing*. <http://ardupilot.org/plane/docs/automatic-landing.html>. [Online; acessado em Junho/2017].
- [8] Gumstix, Inc. *Gumstix Developer Center*. <https://gumstix.org/>. [Online; acessado em Junho/2017].
- [9] BROWN, J. H. How fast is fast enough? choosing between xenomai and linux for real-time applications. 2016.
- [10] ABBOTT, D. *Linux for embedded and real-time applications*. 2. ed. [S.l.]: Newnes, 2006. (Embedded Technology). ISBN 9780750679329,0750679328.
- [11] Xenomai Development Community. *Xenomai Documentation*. https://xenomai.org/start-here/#How_does_Xenomai_deliver_real-time. [Online; acessado em Junho/2017].
- [12] Gipsa-Lab and Institute of Mouvement Sciences. *RT-MaG Project Documentation*. <http://www.gipsa-lab.fr/projet/RT-MaG/index.php>. [Online; acessado em Junho/2017].
- [13] APM Development Community. *MAVLink - Micro Air Vehicle Communication Protocol Documentation*. <http://qgroundcontrol.org/mavlink/start>. [Online; acessado em Junho/2017].

- [14] BALASUBRAMANIAN, S. *MAVLink Tutorial for Absolute Dummies*. http://api.ning.com/files/i*tFWQTF2R*7Mmw7hksA. [Online; acessado em Junho/2017].
- [15] Microhard System Inc. *Pico Series P900 Operating Manual.v1.8.5*. Canada: [s.n.], 2016.