

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARTHUR BENEMANN

**ESTAÇÃO DE CONTROLE PARA VEÍCULOS AÉREOS NÃO
TRIPULADOS**

Porto Alegre
2013

ARTHUR BENEMANN

**ESTAÇÃO DE CONTROLE PARA VEÍCULOS AÉREOS NÃO
TRIPULADOS**

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

2013

ARTHUR BENEMANN

ESTAÇÃO DE CONTROLE PARA VEÍCULOS AÉREOS NÃO TRIPULADOS

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de “Projeto de Diplomação”, do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Formação Universidade Federal do Rio Grande do Sul – Porto
Alegre, Brasil

Banca Examinadora:

Prof. (Nome do professor), sigla da Instituição onde atua

Doutor pela (Instituição onde obteve o título – Cidade, País)

Título (Nome do profissional), sigla da Instituição onde atua

Doutor pela (Instituição onde obteve o título – Cidade, País)

Prof. Dr. (Nome do professor), sigla da Instituição onde atua

Doutor pela (Instituição onde obteve o título – Cidade, País)

Porto Alegre

2013

AGRADECIMENTOS

Agradeço à minha família pelo suporte durante o período da graduação. Gostaria de agradecer também à Universidade Federal do Rio Grande do Sul que ofereceu uma das melhores graduações de engenharia elétrica do Brasil.

Aos colegas do LASCAR pelo auxílio nas tarefas desenvolvidas na confecção e revisão deste trabalho.

"Free software is a matter of liberty, not price. To understand the concept, you should think of
free as in free speech, not as in free beer."

Richard Stallman

RESUMO

Os avanços tecnológicos relacionados a motores elétricos de pequeno porte e baterias de alta densidade energética possibilitaram que os micro veículos aéreos não tripulados (VANT) ganhassem foco na comunidade científica. Este trabalho propõe a criação de uma estação de controle (Ground Control Station) para facilitar a interação com os sistemas quando operados em campo, com o objetivo de facilitar o planejamento de missões a serem executadas pelo VANT e acompanhar os dados de voo enquanto esta é realizada.

Palavras-chaves: Engenharia Elétrica. VANT. GCS.

Abstract

Technological advancements related to small electrical engines, and high-density batteries, made micro unmanned aerial vehicles (UAV) earn a focus on the scientific community. This paper proposes the creation of a ground control station (GCS) to facilitate interaction with the systems when operated in the field. In order to facilitate the planning of missions to be performed by UAVs, and monitor flight data while it is executed.

Keywords: Electrical Engineering. UAV. GCS.

LISTA DE ILUSTRAÇÕES

Figura 1 - 3DRobotics ArduCopter Y6.....	14
Figura 2 - Módulo Xbee-XSC SMA.....	16
Figura 3 - Módulo HM-TRP.....	17
Figura 4 - Módulo RN-42.....	18
Figura 5 - Módulo RN-171.....	19
Figura 6 - Logo do sistema operacional Android.....	20
Figura 7 - Voo fotogramétrico.....	23
Figura 8 - Nexus 7.....	25
Figura 9 - 3DR ArduCopter Quad C Frame.....	26
Figura 10 - Componentes do quadcoptero.....	26
Figura 11 - Diagrama do link de comunicação da placa MAVBridge.....	28
Figura 12 - MAVBridge – Diagrama de blocos.....	29
Figura 13 - Módulo Bluetooth/Wifi.....	30
Figura 14 - Módulo HopeRF.....	30
Figura 15 - Carregador de bateria LiPo.....	31
Figura 16 - MAVBridge – Fonte Chaveada.....	33
Figura 17 - MAVBridge – Alarme de bateria baixa.....	33
Figura 18 - MAVBridge – Layout dos componentes.....	34
Figura 19 - MAVBridge – Layout da camada superior.....	35
Figura 20 - MAVBridge – Layout da camada inferior.....	35
Figura 21 - Radio Hope RF conectado via porta USB através de um adaptador.....	36
Figura 22 - módulo de telemetria 3DRobotics.....	36
Figura 23 - Interação típica dos componentes do modelo MVC.....	39
Figura 24 - Diagrama de blocos da arquitetura do software.....	40
Figura 25 - Diagrama de blocos da interface de usuário.....	40
Figura 26 - Diagrama de blocos do VANT virtual.....	42
Figura 27 - Diagrama de blocos da interface de comunicação.....	44
Figura 28 - Tela de Informações de voo.....	45
Figura 29 - Detalhes do Heads Up Display.....	46
Figura 30 - Tela de Planejamento de missões.....	48
Figura 31 - Tela de Planejamento de missões.....	49
Figura 32 - Tela de Planejamento aerofotogramétrico.....	50

Figura 33 - Tela de Controle – visualização horizontal (paisagem).....	51
Figura 34 - Tela de Controle – visualização vertical (retrato).....	51
Figura 35 - Tela de Parâmetros.....	53
Figura 36 - Exemplo de tela de detalhe de parâmetros.....	53
Figura 37 - Tela de Gráficos.....	54
Figura 38 - número de dispositivos com o aplicativo instalado.....	56
Figura 39 - Exemplo de resultados fotogramétricos.....	57

LISTA DE ABREVIATURAS

ASCII:	<i>American Standard Code for Information Interchange</i>
CRC:	<i>Cyclic Redundancy Check</i>
GCS:	<i>Ground Control Station</i>
HUD	<i>Heads Up Display</i>
IHM:	Interface Homem Maquina
LASCAR:	Laboratório de Sistemas de Controle, Automação e Robótica
LiPo:	<i>Lithium Polymer Battery</i>
LSB:	<i>Least Significant Bits</i>
MAVLink:	<i>Micro Air Vehicle Link</i>
MSB:	<i>Most Significant Bits</i>
PCB:	<i>Printed Circuit Board</i>
PCI:	Placa de Circuito Impresso
RF:	Rádio Frequência
RTF	<i>Ready To Fly</i>
TCP:	<i>Transmission Control Protocol</i>
UAV:	<i>Unmanned Aircraft Vehicle</i>
UDP:	<i>User Datagram Protocol</i>
UVLO:	<i>UnderVoltage LockOut</i>
VARP:	Veículo Aéreo Remotamente Pilotado
VANT:	Veículo Aéreo Não Tripulado
XML:	<i>Extensible Markup Languages</i>

SUMÁRIO

1	INTRODUÇÃO	13
2	REVISÃO DE CONCEITOS	14
2.1	VEÍCULOS AÉREOS NÃO TRIPULADOS.....	14
2.1.1	Plataforma ArduCopter.....	14
2.2	ESTAÇÃO DE CONTROLE DE SOLO.....	15
2.2.1	Dados de voo.....	15
2.2.2	Controle.....	15
2.2.3	Planejamento.....	15
2.3	REDES SEM FIO.....	16
2.3.1	Xbee.....	16
2.3.2	Hope-RF.....	17
2.3.3	Bluetooth.....	17
2.3.4	WiFi.....	18
2.4	PLATAFORMA ANDROID.....	19
2.5	PROTÓCOLO DE COMUNICAÇÃO.....	20
2.5.1	Pacotes MAVLink.....	21
2.5.2	CRC.....	21
2.5.3	Mensagens.....	21
2.6	AEROFOTOGRAMETRIA.....	22
3	HARDWARE	24
3.1	ESTAÇÃO DE CONTROLE.....	24
3.2	VANT.....	25
3.3	LINK DE COMUNICAÇÃO.....	27
3.3.1	MAVBridge.....	28
3.3.2	Conexão direta via USB.....	35
4	SOFTWARE	38
4.1	ARQUITETURA.....	38
4.1.1	Interface de usuário.....	40
4.1.2	VANT virtual.....	42
4.1.3	Meios de Comunicação.....	44
4.2	INTERFACE HOMEM MÁQUINA.....	45
4.2.1	Informações de voo.....	45

4.2.2	Planejamento de missões.....	48
4.2.3	Controle.....	51
4.2.4	Parâmetros.....	53
4.2.5	Gráficos.....	54
5	RESULTADOS	56
6	PROPOSTAS DE MELHORIAS	58
7	CONCLUSÕES	59
	REFERÊNCIAS	60
	APÊNDICE A – ESQUEMÁTICO MAVBRIDGE	62

1 INTRODUÇÃO

O avanço de diversas tecnologias relacionadas a micro veículos aéreos não tripulados (VANT) fizeram com que o seu potencial para uso civil fosse elevado, considerando que a maior parcela de uso, atualmente, encontra-se em fins militares. Como são sistemas versáteis, as possibilidades de uso variam incluindo, principalmente, monitoramento de linhas de transmissão, apoio à segurança pública e agricultura de precisão.

Por serem sistemas complexos, existe grande dificuldade em sua operação. Este trabalho propõe a criação de uma estação de controle (*Ground Control Station*, GCS) para facilitar a interação com os VANTS quando operados em campo, facilitando o planejamento e execução de missões do sistema além de acompanhar os dados de voo de forma simultânea. A implementação do projeto terá como base a plataforma *Android*, através da elaboração de um aplicativo.

Posteriormente, serão apresentados de forma mais detalhada o problema e a elaboração da solução, incluindo o protocolo de comunicação utilizado, a implementação do sistema e os resultados de experimentos práticos realizados para a validação do projeto.

2 REVISÃO DE CONCEITOS

2.1 VEÍCULOS AÉREOS NÃO TRIPULADOS

Um veículo aéreo não tripulado ou veículo aéreo remotamente pilotado (VARP) é todo e qualquer tipo de aeronave que não necessita de pilotos embarcados para ser guiada. Esses aviões são controlados a distância por meios eletrônicos e computacionais, sob a supervisão e governo humanos ou sem a sua intervenção.

A terminologia VANT se refere não somente a aeronave, mas a todos os equipamentos de suporte utilizados no sistema, incluindo sensores, microcontroladores, software, estações de controle de solo, hardware de comunicação. (BEARD;McLAIN, 2012)

Pelo fato de o veículo não ser tripulado todos os esforços de desenvolvimento de uma Interface Homem Maquina (IHM) são deslocados do interior da aeronave para a estação de controle de solo.

2.1.1 Plataforma ArduCopter

Figura 1 - 3DRobotics ArduCopter Y6



Fonte: 3DRobotics, 2013

2.2 ESTAÇÃO DE CONTROLE DE SOLO

A estação de controle é uma parte indispensável em um sistema de VANT. O computador é núcleo de uma GCS, e o software do computador, ou software da GCS, é o seu elemento mais critico. (KANG;YUAN, 2009)

Os objetivos de uma GCS são divididos em visualização de dados de voo, controle da aeronave, planejamento de missões autônomas.

2.2.1 Dados de voo

A percepção da situação, isto é, a habilidade de perceber e monitorar o ambiente nas proximidades da aeronave, é de extrema importância na operação de veículos aéreos. A visualização dos dados de voo como, por exemplo, altitude atual, velocidade do ar, velocidade de solo, orientação espacial, posição espacial, localização geográfica, modo de controle/voo, altitude e velocidade desejados, situação da missão, consumo de energia, é necessária para a construção desta percepção de situação pelo operador do sistema na GCS, dado que não há piloto embarcado.

2.2.2 Controle

O controle da aeronave pode ser realizado de forma direta, modificando diretamente a orientação da aeronave, ou de forma indireta, através de modificações do plano de voo (o qual é seguido de forma autônoma pelo VANT).

2.2.3 Planejamento

Os VANTs são capazes de realizar, de forma autônoma, “missões”, isto é, planos de voo definidos pelo operador. Uma forma de definir uma missão, ou seja, planejar um voo, é através da definição de *waypoints* - pontos tridimensionais os quais devem ser atingidos pela aeronave.

Existem diversos tipos de *waypoints*, dependendo do software de piloto automático utilizado, como por exemplo *waypoints* pontuais, de decolagem, de pouso, de espera, de retorno à casa (ponto de origem do voo, definido pelo operador).

2.3 REDES SEM FIO

Uma rede sem fio (ou comunicação sem fio) refere-se a uma passagem aérea sem a necessidade do uso de cabos – sejam eles telefônicos, coaxiais ou ópticos – por meio de equipamentos que usam radiofrequência (comunicação via ondas de rádio).

No contexto dos VANTs as redes sem fio são utilizadas para criar o link de telemetria do veículo, de forma a ser possível controlar/observar dados do veículo em operação remotamente. Algumas das limitações apresentadas por essa aplicação são a baixa potência disponível, limitações de espaço físico e as distâncias relativamente grandes a serem superadas.

2.3.1 XBee

XBee é a marca da Digi International para uma família de módulos de rádio compatíveis entre si quanto as dimensões físicas. Os primeiros rádios XBee foram introduzidos sob a marca MaxStream em 2005 e foram baseadas no padrão 802.15.4-2003 projetado para comunicações estrela ponto-a-ponto e em taxas de transmissão de 250 kbit/s. A potência de transmissão varia de 1mW a 100mW, dependendo do módulo utilizado. A figura 2 apresenta um destes módulos (XBee-XSC SMA). A possibilidade da implementação de redes de sensores é interessante no contexto de VANTs, podendo ser utilizada para a criação de redes de grande extensão.

Figura 2 - Módulo XBee-XSC SMA

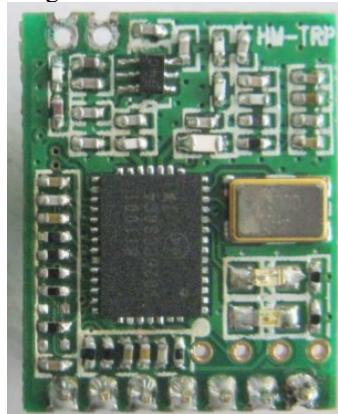


FONTE: DIGI, 2013

2.3.2 Hope-RF

Série HM-TRP da empresa Hope Microelectronics são rádios de baixo custo, alta performance com a operação em 433/470/868/915 MHz. A potencia de saída é de 100mW, com uma sensibilidade de -117dBm, o que possibilita um longo alcance. A taxa de comunicação é de 250 kbits/s (HOPE MICROELECTRONICS). A figura 3 exibe um destes módulos.

Figura 3 - Módulo HM-TRP



FONTE: HOPE MICROELECTRONICS, 2013

Um projeto *open-source* chamado SiK foi criado para aumentar as possibilidades deste hardware (TRIDGELL). Uma das vantagens deste novo *firmware* é a interligação entre os pacotes de radio e os pacotes do protocolo MAVLink, o que reduz o numero de pacotes perdidos devido a fragmentação.

2.3.3 Bluetooth

Bluetooth é uma especificação industrial para áreas de redes pessoais sem fio (*Wireless personal area networks*). O Bluetooth provê uma maneira de conectar e trocar informações entre dispositivos como telefones celulares, notebooks, computadores, impressoras, câmeras digitais e consoles de videogames digitais através de uma frequência de

rádio de curto alcance globalmente licenciada e segura. As especificações do Bluetooth foram desenvolvidas e licenciadas pelo "Bluetooth Special Interest Group". É um protocolo padrão de comunicação primariamente projetado para baixo consumo de energia com baixo alcance, baseado em microchips transmissores de baixo custo em cada dispositivo.

Devido ao curto alcance desta tecnologia, quando comparados com distâncias de operação de VANTs, eles tem pouca aplicação para os propósitos deste trabalho além de configuração em solo dos veículos. Porem conforme explicitados no capítulo 3.3.1, devido ao fato de este dispositivo já estar integrado com um grande numero de dispositivos Android ele pode ser uma alternativa interessante para uma rede local.

A figura 4 apresenta um módulo produzido pela empresa Microchip que integra toda a funcionalidade necessária para a utilização deste protocolo. Um produto derivado ,da mesma empresa, contem o modulo montado em uma placa de circuito impresso com as mesmas dimensões e pinagem de um modulo Xbee, possibilitando a fácil troca de módulos.

Figura 4 - Módulo RN-42



FONTE: MICROCHIP, 2013

2.3.4 WiFi

Wi-Fi é uma marca registrada da Wi-Fi Alliance. É utilizada por produtos certificados que pertencem à classe de dispositivos de rede local sem fios (WLAN) baseados no padrão IEEE 802.11. Atualmente, praticamente todos os computadores portáteis vêm de fábrica com dispositivos para rede sem fio no padrão Wi-Fi (802.11b, a, g ou n). O que antes era acessório está se tornando item obrigatório, principalmente devido ao fato da redução do custo de fabricação.

A frequência de operação é de 2,4 ou 5,8 GHz (dependendo do padrão), com a potencia máxima de operação variando de 1W a 10mW (dependendo da legislação do país

onde é operado). As principais vantagens deste protocolo são a alta taxa de comunicação (10Mbps a 600Mbps) e a disponibilidade de mecanismos de segurança (WEP, WPA, WPA2). Porem devido a uma das desvantagens, o baixo alcance, ele é pouco atrativo para a aplicação de VANTS (pelos mesmos motivos do protocolo Bluetooth).

O modulo RN-171 produzido pela empresa Microchip foi utilizado neste projeto, pois já contem todo o stack necessário para realizar a comunicação WiFi. A figura 5 apresenta este modulo. Assim como no caso do modulo Bluetooth (RN-42) uma versão com o mesmas dimensões e pinagens de um Xbee é disponibilizada.

Figura 5 - Módulo RN-171



FONTE: MICROCHIP, 2013

2.4 PLATAFORMA ANDROID

Android é um sistema operacional baseado no núcleo do Linux, mas destinado a dispositivos móveis. Desenvolvido pela *Open Handset Alliance*, o sistema teve grande popularização desde seu ano de lançamento público, em 2008, dado que incorpora uma combinação de facilidade de uso com disponibilidade gratuita. Juntamente com o sistema operacional, observou-se o crescimento da utilização dos aplicativos, isto é, softwares destinados a uma tarefa específica, sendo, assim, “ferramentas” do sistema.

Majoritariamente, o desenvolvimento de aplicativos se dá através de programação em Java, uma linguagem de programação orientada ao objeto e com sintaxe similar às linguagens C/C++. Para o projeto descrito neste documento, a linguagem de programação utilizada foi Java. A figura 6 apresenta o logotipo do sistema operacional.

Figura 6 - Logo do sistema operacional Android



FONTE: Google inc.

2.5 PROTOCOLO DE COMUNICAÇÃO

O protocolo de comunicação utilizado foi o MAVLink v1.0, sendo escolhido pois existe um grande suporte em VANTS já existentes, como: Ardupilot, PX4FMU, SmartAP, MatrixPilot. Isso possibilita a fácil integração com todos estes sistemas.

O protocolo foi projetado por Lorenz Meier [\[1\]](#), e liberado para uso sobre a licença [LGPL](#) em 2009. O uso mais comum é para a comunicação de um VANT para a sua GCS, e na intercomunicação de subsistemas do veículo. Através deste protocolo, dados do VANT, como posição e orientação, são transferidos.

A comunicação dos dados é realizada através de pacotes, os quais contêm uma estrutura parecida com o protocolo [CAN](#). Cada pacote contém uma mensagem de tamanho variável, cujos dados dependem da mensagem a ser transmitida. A ordem dos dados contidos em cada mensagem é definida em um arquivo XML, que faz parte do protocolo. VANTS de fabricantes diferentes podem estender o conjunto de mensagens padrão, oferecendo certa flexibilidade, embora isso possa o tornar incompatível com as GCS disponíveis.

2.5.1 Pacotes MAVLink

A anatomia dos pacotes é apresentada na tabela 1, esta estrutura é baseada nos protocolos CAN e SAE AS-4. O foco do desenvolvimento deste protocolo foi a velocidade de transmissão e segurança. Com apenas seis bytes extras impostos nas mensagens é possível realizar a verificação dos dados da mensagem e detectar a perda de pacotes.

Tabela 1 Estrutura de um pacote MAVLink v1.0

Nome do Campo	Posição (Bytes)	Propósito
Início de pacote	0	Denota o início de um pacote (v1.0 0xFE)
Tamanho da mensagem	1	Tamanho em bytes da mensagem seguinte
Número de sequência	2	Número do pacote na sequência
ID do Sistema	3	Identificação do sistema que está enviando este pacote
ID do Componente	4	Identificação do componente que está enviando este pacote
ID da mensagem	5	Define o que a mensagem contém, e como ela deve ser decodificada
Mensagem	6 a (n+6)	Dados dessa mensagem, dependente do ID da mensagem
CRC (<i>Cyclic Redundancy Check</i>)	(n+7) a (n+8)	<i>Checksum</i> do pacote, para detecção de erros de transmissão

2.5.2 CRC

Para garantir a integridade das mensagens, um campo contendo o *checksum* é adicionado ao final de cada pacote. Na recepção, o *checksum* é recalculado e comparado com o recebido, e caso ocorra uma diferença a mensagem é descartada.

Outra função do campo de CRC é garantir que o transmissor e o receptor concordem com o tipo de dado sendo transferido. Isso é feito adicionando uma “semente” (um valor inicial) ao gerador de *checksum* baseada em uma função *hash* da definição da mensagem. Normalmente um vetor estático é gerado com o valor de *hash* correspondente a cada mensagem, diminuindo o custo computacional para um sistema *real-time*.

2.5.3 Mensagens

Os dados transmitidos são contidos em mensagens estruturadas, as quais posteriormente são encapsuladas pelos pacotes já descritos na seção anterior. Cada mensagem é identificada pelo campo ID de mensagem do pacote, e tem os seus dados armazenados na seção de carga do pacote.

A decodificação/codificação de cada mensagem é feita de acordo com um arquivo XML definido pelo protocolo. Este arquivo contém a ordem dos campos a serem extraídos, o tamanho de cada campo e o tipo de dado armazenado.

Na ... abaixo é apresentada a definição de uma mensagem típica do protocolo, extraída do documento XML.

```
<message id="24" name="GPS_RAW_INT">
    <description>The global position, as returned by the Global Positioning System (GPS). This is NOT the
    global position estimate of the system, but rather a RAW sensor value. See message GLOBAL_POSITION for the
    global position estimate. Coordinate frame is right-handed, Z-axis up (GPS frame).</description>
    <field type="uint64_t" name="time_usec">Timestamp (microseconds since UNIX epoch or microseconds
    since system boot)</field>
    <field type="uint8_t" name="fix_type">0-1: no fix, 2: 2D fix, 3: 3D fix. Some applications will not use the
    value of this field unless it is at least two, so always correctly fill in the fix.</field>
    <field type="int32_t" name="lat">Latitude (WGS84), in degrees * 1E7</field>
    <field type="int32_t" name="lon">Longitude (WGS84), in degrees * 1E7</field>
    <field type="int32_t" name="alt">Altitude (WGS84), in meters * 1000 (positive for up)</field>
    <field type="uint16_t" name="eph">GPS HDOP horizontal dilution of position in cm (m*100). If unknown,
    set to: UINT16_MAX</field>
    <field type="uint16_t" name="epv">GPS VDOP horizontal dilution of position in cm (m*100). If unknown,
    set to: UINT16_MAX</field>
    <field type="uint16_t" name="vel">GPS ground speed (m/s * 100). If unknown, set to:
    UINT16_MAX</field>
    <field type="uint16_t" name="cog">Course over ground (NOT heading, but direction of movement) in
    degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX</field>
    <field type="uint8_t" name="satellites_visible">Number of satellites visible. If unknown, set to
    255</field>
</message>
```

2.6 AEROFOTOGRAMETRIA

Aerofotogrametria é a cobertura aerofotográfica executada para fins de mapeamento. O uso de um conjunto de fotos tiradas a uma altura predeterminada do solo pode ser utilizada para gerar um mosaico do terreno, sendo possível, inclusive, geo-referenciar este mosaico para determinar as dimensões e posições de objetos contidos nele.

Para a obtenção de um conjunto satisfatório de fotos para geração do mosaico é necessário que exista uma sobreposição longitudinal e lateral entre as imagens. Isso é obtido através de um planejamento de voo cuidadoso, conforme mostrado na figura 7.

Figura 7 - Voo fotogramétrico



Fonte: UFF, 2013

3 HARDWARE

O objetivo deste projeto é o desenvolvimento de uma estação de controle de solo para VANTs, com o foco em pequenos veículos. Normalmente estes veículos são operados em linha de visão, com pouco tempo para configurar uma base de comando, por isso existem as seguintes metas de projeto:

- Portátil
- Visualização de dados de voo em tempo real
- Planejamento de missões autônomas
- Controle do VANT
- Possibilidade de configurar parâmetros do VANT
- Baixo peso
- Baixo custo

Para facilitar o projeto do sistema, este foi dividido em: estação de controle, VANT e link de telemetria. Cada item será detalhado nos capítulos seguintes.

3.1 ESTAÇÃO DE CONTROLE

A estação de controle é o equipamento que será utilizado em campo para o controle, visualização, e planejamento do voo do VANT. Para atender a todas as especificações de projeto, e buscando reduzir o custo do sistema, optou-se pela utilização de dispositivos Android disponíveis no mercado. Por serem um produtos os quais já estão amplamente difundidos comercialmente, eles apresentam as seguintes vantagens:

- Programação simples
- Alta disponibilidade
- Baixo custo
- Alto poder computacional em relação ao preço
- Baixo peso
- Portabilidade

Selecionou-se o *tablet* Nexus 7 (2013) da empresa Google por atender os requisitos desta aplicação, além do fato de ser considerado o dispositivo de referência para o sistema Android. Na figura 8 é apresentado a visão frontal do Nexus 7.

Figura 8 - Nexus 7



FONTE: Google inc.

Como a única restrição deste projeto quanto a estação de controle é de que o *hardware* seja compatível com o sistema operacional Android, a gama de dispositivos disponíveis é grande. Alguns dos dispositivos nos quais o *software* foi utilizado com sucesso são: Nexus 7 (2012), Nexus 5, Nexus 4, Nexus 10, Asus TF300T e TF300TG, Samsung Galaxy Note 2, Samsung Galaxy Note 3, Samsung Galaxy Tab 2 7.0, Samsung Galaxy Tab 10.1, Samsung Galaxy S3, Samsung Galaxy S4, Samsung Galaxy Nexus, Xperia Z e Z1, Genesis GT-7230, T-pad tablet IS701 e IS709C, Acer Iconia A500, A501 e A510 (Fonte: *Droidplanner - Compatible devices*).

3.2 VANT

Este projeto suporta o uso de diversos tipos de VANT, como veículos multi-rotoreis e asa-fixa, porém, para a realização de experimentos foi construído um veículo do tipo asa fixa. Um kit de quadricoptero (multi-rotor com 4 motores) da empresa 3DRobotics foi utilizado. O sistema de controle *on-board* é o ArduPilot, um projeto

open-source para controle de VANTs, compatível com o protocolo MAVLink. Uma foto do protótipo montado é apresentada na figura 9.

Figura 9 - 3DR ArduCopter Quad C Frame



Os componentes utilizados na montagem deste quadricoptero estão listados abaixo, e apresentados na figura 10.

- ArduCopter 3DR Quad Frame KIT
- GPS uBlox LEA-6
- Módulo de potência 3DR APM
- Electronic Speed Controller (ESC) 20A 3DR
- Hélices APC 10"x4,7"
- Telemetria Hope-RF 915 MHz

Figura 10 - Componentes do quadcoptero



3.3 LINK DE COMUNICAÇÃO

O link de comunicação é vital para o funcionamento correto da estação de controle, pois é através dele que são transmitidos e recebidos todos os dados da aeronave. Como a aplicação deste projeto é para a operação de pequenos VANTs, os objetivos de projeto são:

- Link de comunicação estável
- Taxa de transmissão de aproximadamente 57 kbits/s (valor necessário para obter uma taxa de atualização aceitável dos dados do VANT)
- Baixo consumo energético, pois existe a limitação das baterias, tanto no VANT quanto na Estação de controle
- Pequenas dimensões, dadas as limitações físicas e a portabilidade da estação de controle, e também o tamanho dos VANTs

Como o objetivo do projeto é o desenvolvimento da estação de controle, buscou-se soluções comercialmente disponíveis. Os links de rádio digitais que foram encontrados são: Xbee, HopeRF, WiFi e Bluetooth (detalhados no capítulo de revisão de conceitos).

O link de Xbee foi utilizado inicialmente, entretanto acabou sendo descartado devido a baixa performance nos experimentos. Uma das falhas encontradas é o grande tempo

de latência para a entrega de pacotes de rádio quando o sinal se torna marginal, o que torna esse link inviável para o controle em tempo real do veículo. Possivelmente, uma implementação de rede de sensores, incluindo sensores móveis em VANTs, eliminaria as limitações. Estudos sobre este problema podem ser encontrados em (PIGNATON) e (PEREIRA).

O sistema baseado no *transceiver* Si1000 da empresa HopeRF apresentou resultados superiores em se tratando de qualidade do link e alcance, quando comparados aos obtidos com o Xbee. Um dos fatores que contribuem para a performance deste rádio é o fato de utilizar um *firmware* desenvolvido especificamente para transmissão de dados no protocolo MAVLink, cujo projeto se encontra em (SiK). Para a conexão deste rádio com o dispositivo Android foi utilizada a porta USB disponível no módulo projetado pela empresa 3DRobotics. Um problema foi encontrado com a utilização do Si1000, que é a necessidade de ter a antena conectada fisicamente no *tablet* (reduzindo a portabilidade do projeto).

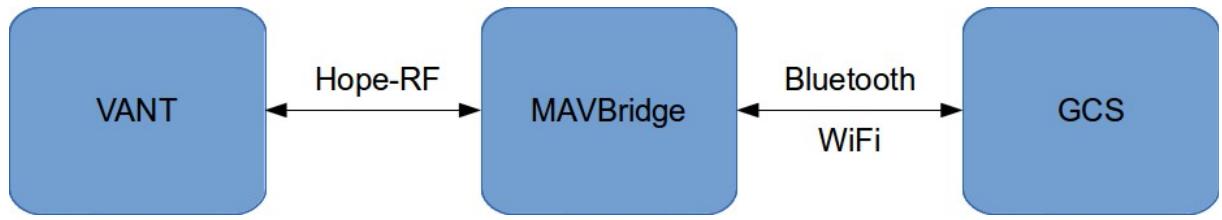
As soluções baseadas em WiFi e Bluetooth foram descartadas inicialmente por não terem o alcance necessário para esta aplicação. Porém, elas são úteis no caso de se utilizar a estação de controle apenas para a configuração de voos autônomos no VANT.

Nenhum dos rádios analisados anteriormente atingiu todas as metas definidas, portanto foi projetado um *hardware* para utilizar os pontos fortes de cada tipo de rádio, sendo chamado de MAVBridge.

3.3.1 MAVBridge

Devido aos problemas explicitados anteriormente foi desenvolvida uma placa para utilizar dois tipos de rádio. Os rádios utilizados são o módulo da HopeRF, por sua performance quando utilizado no link VANT-Solo, e o link de Bluetooth, que apesar de ter um alcance pequeno, possibilita o uso do *tablet* sem nenhum aparelho conectado a ele fisicamente. Como alternativa ao Bluetooth é possível utilizar WiFi nessa placa. A figura 11 ilustra o funcionamento dessa placa mostrando os três componentes envolvidos no link.

Figura 11 - Diagrama do link de comunicação da placa MAVBridge

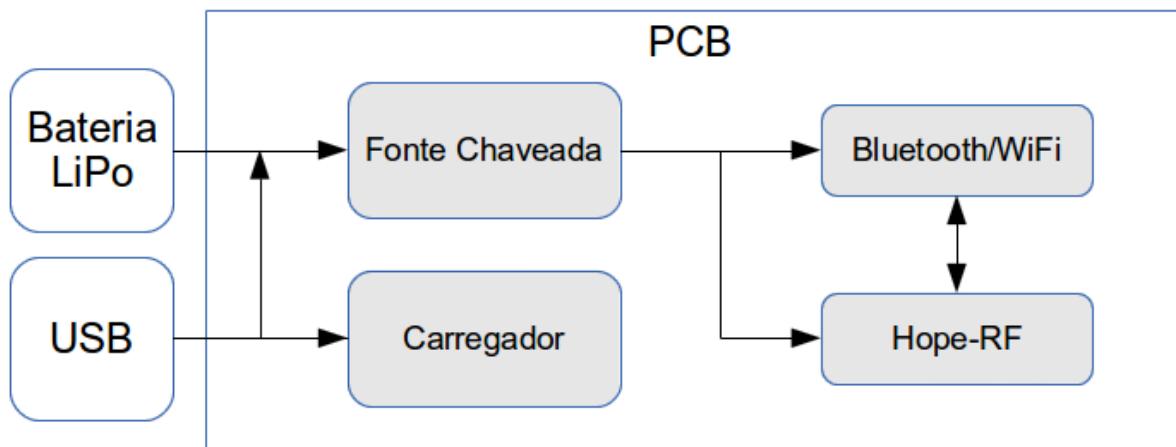


Algumas das vantagens quanto a utilização desta placa são:

- Possibilidade de colocar a antena de telemetria em uma localização mais privilegiada (melhorando a integridade do sinal de RF)
- Não é necessário nenhum *hardware* extra conectado ao dispositivo Android
- Aumento da autonomia do sistema, pois a energia utilizada pelos rádios é obtida de uma bateria dentro da MAVBridge, distribuindo, assim, a carga entre as baterias do dispositivo Android e da MAVBridge

A placa pode ser dividida em quatro partes conforme a figura 12. Existe uma conexão direta entre ambos os módulos de rádio e um sistema de alimentação e recarga da bateria de *Lithium Polymer* (LiPo).

Figura 12 - MAVBridge – Diagrama de blocos



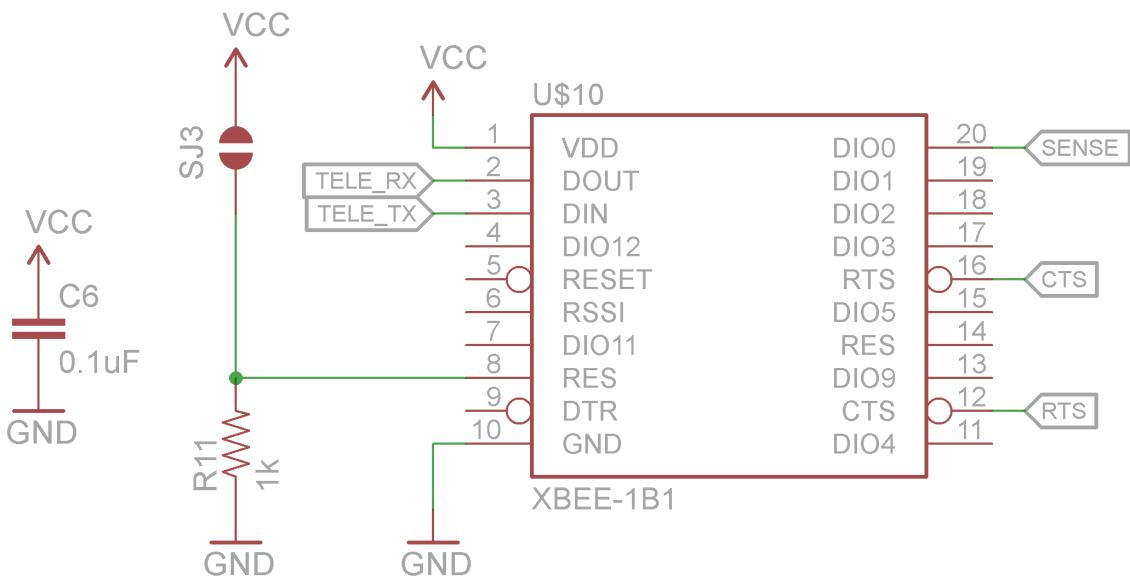
3.3.1.1 Esquemático

O projeto da MAVBridge é relativamente simples pois o *design* dos circuitos de Rádio Frequência (RF) já estão prontos, devido ao amplo uso dos módulos da Microchip.

Entretanto, o layout deve ser realizado de forma correta para evitar ruídos na fonte de alimentação. O esquemático completo encontra-se no Apêndice A desse documento.

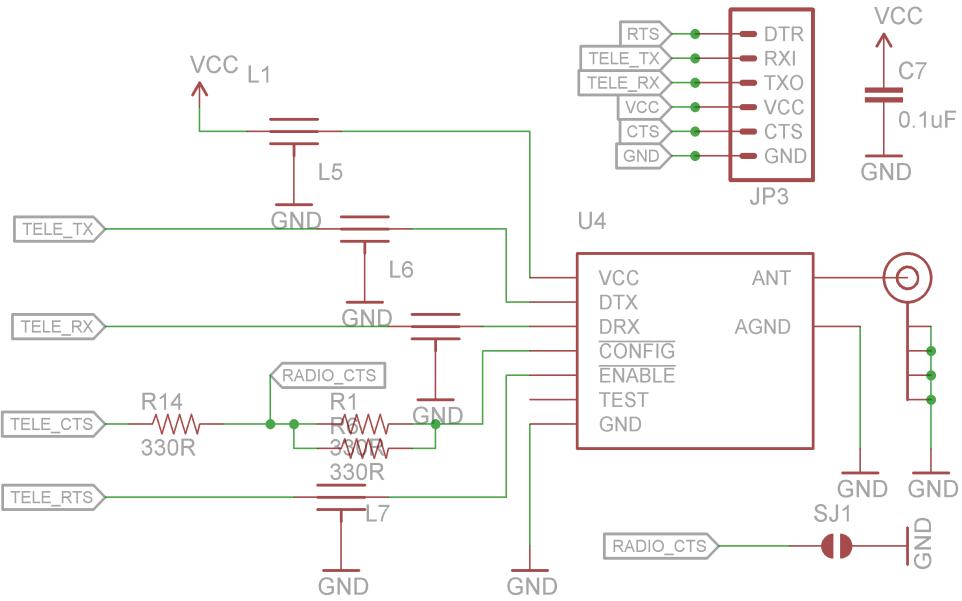
A figura 13 mostra a parte do esquemático responsável pelos módulos Bluetooth ou WiFi. Como os módulos RN-42 e RN-171 da empresa Microchip (figuras 4 e 5) tem as mesmas pinagem, dimensões e funcionalidade, é possível apenas trocar o módulo que está instalado na placa, possibilitando o uso tanto de WiFi como de Bluetooth para o link local. O *jumper* SJ3 serve para reiniciar o módulo com os padrões do fabricante.

Figura 13 - Módulo Bluetooth/Wifi



A figura 14 contém o módulo de rádio da empresa HopeRF, e o layout é baseado nos módulos de telemetria da empresa 3DRobotics. Os filtros L1, L5, L6 e L7 são utilizados para reduzir o *slew-rate* dos sinais digitais entrando no módulo e para filtrar a linha de alimentação. Uma alimentação sem ruído é essencial para o funcionamento correto deste módulo. O *jumper* SJ1 em conjunto com os resistores R14, R1 e R6 pode colocar o módulo em modo de *bootloader*, possibilitando a mudança do *firmware* utilizado. O conector JP3 pode ser utilizado caso seja necessário utilizar um módulo diferente do Hope-RF para a saída de telemetria.

Figura 14 - Módulo HopeRF

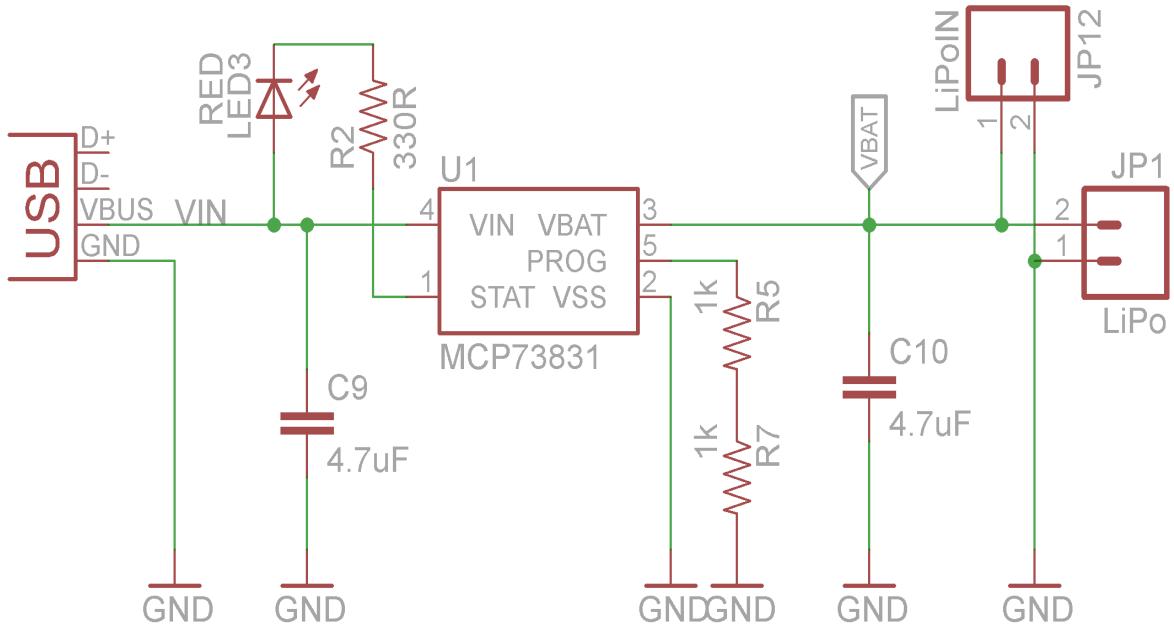


A bateria utilizada no projeto é uma do tipo LiPo, que requer alguns cuidados especiais para ser realizada a carga. Mas como vantagem ela apresenta uma alta densidade energética e um custo baixo, quando comparada a outras baterias semelhantes.

O ciclo de carga é realizado pelo circuito da figura 15, que é a implementação típica do Circuito Integrado (CI) MCP73831 (MICROCHIP). A corrente de carga programada pelos resistores R7 e R5 é de 500mA, o que é seguro para a bateria utilizada (capacidade de 2200mAh) e fornece uma carga rápida (aproximadamente 5h). O componente LED3 indica que a bateria está sendo carregada.

A alimentação externa é provida por um conector USB micro-B, o qual deve estar conectado a uma fonte de 5V. A utilização deste conector, que é o mesmo utilizado na maioria dos dispositivos Android, significa que não é necessário o desenvolvimento de uma fonte externa, uma vez que o usuário já deve possuir uma fonte adequada (a da própria estação de controle), ou pode utilizar a porta USB de um computador pessoal.

Figura 15 - Carregador de bateria LiPo



A fonte chaveada tem a função de regular a tensão de alimentação do circuito, utilizando a energia armazenada na bateria. Entretanto, a utilização de uma bateria do tipo LiPo gera algumas peculiaridades no projeto desta fonte de alimentação.

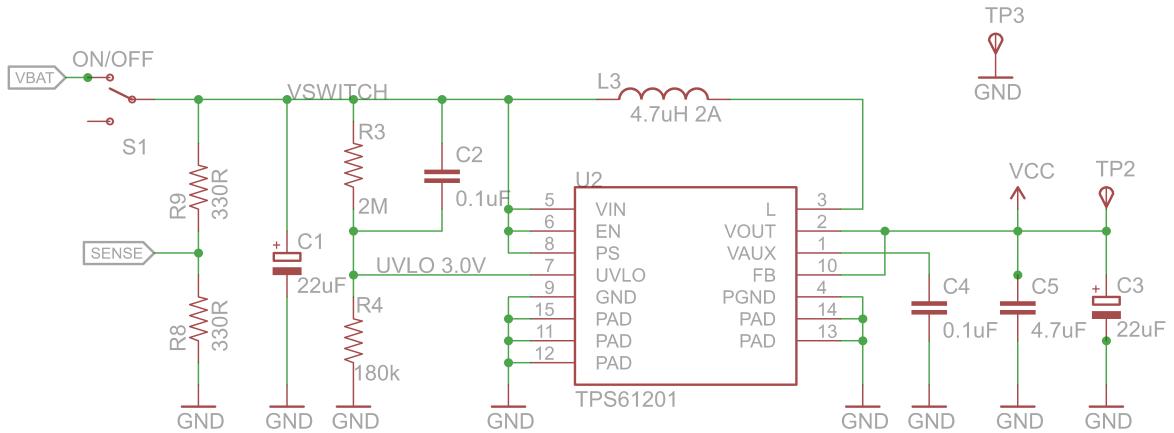
O problema encontrado é que a tensão de alimentação do circuito é fixa em 3,3V, enquanto a tensão da bateria varia entre 4,2 V a 2,75 V (dependendo principalmente da quantidade de carga restante). É necessário a utilização de uma fonte chaveada (para manter a eficiência elevada), porém esta tem que ser capaz de operar conforme duas topologias. Quando a tensão da bateria é menor do que a do circuito ela deve operar como *boost*, e caso contrário, deve operar como *buck*. A topologia *buck-boost* não é aplicável pois ela inverte a tensão de saída.

Outra dificuldade é que as baterias LiPo não podem ser descarregadas abaixo da tensão de 2,75V, pois isso danifica a bateria. Dessa forma é necessário um circuito de proteção contra sub-tensão na fonte chaveada.

Essas restrições de projeto foram solucionadas pelo uso do C.I. TPS61201 (TEXAS INSTRUMENTS), indicado especialmente para esse uso (alimentação de circuitos digitais a partir de baterias LiPo), e que muda internamente a topologia de funcionamento entre *buck* e *boost*. O esquemático da figura 16 é o circuito de aplicação típica do *datasheet*, apenas com a modificação do circuito de *UnderVoltage LockOut* (UVLO), para que essa

proteção atue com tensões de 3V. Um divisor de tensão formado por R9 e R8 serve de ponto de medida de tensão da bateria, podendo ser utilizado pelos módulos de WiFi/Bluetooth para estimar a carga restante. A chave S1 serve de interruptor para o circuito, porém deixa o circuito de carga conectado a bateria, para que seja possível carregar enquanto o circuito está desligado.

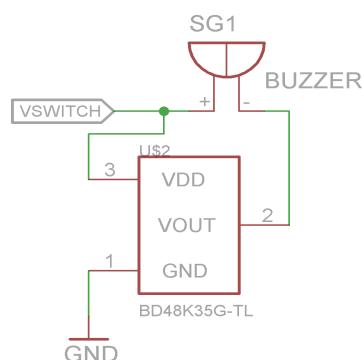
Figura 16 - MAVBridge – Fonte Chaveada



O risco de danificar a bateria devido a descarga extrema foi resolvido pelo circuito de UVLO, entretanto, é necessário uma indicação de que a bateria está próxima deste ponto. Uma indicação sonora foi projetada para operar quando a tensão atinge o nível de 3,5 V, o que representa que existem apenas 10% de carga restante na bateria.

A detecção de nível é realizada pelo C.I. BD48K35, o qual é utilizado normalmente para proteção de UVLO de circuitos digitais, porém foi reutilizado aqui devido as suas características interessantes. Ele contém uma referência de tensão estabilizada internamente, e um comparador com saída *open-collector* de alta corrente de saída, capaz de acionar diretamente um *buzzer* piezoelétrico. A figura 17 mostra o circuito projetado.

Figura 17 - MAVBridge – Alarme de bateria baixa



3.3.1.2 Layout

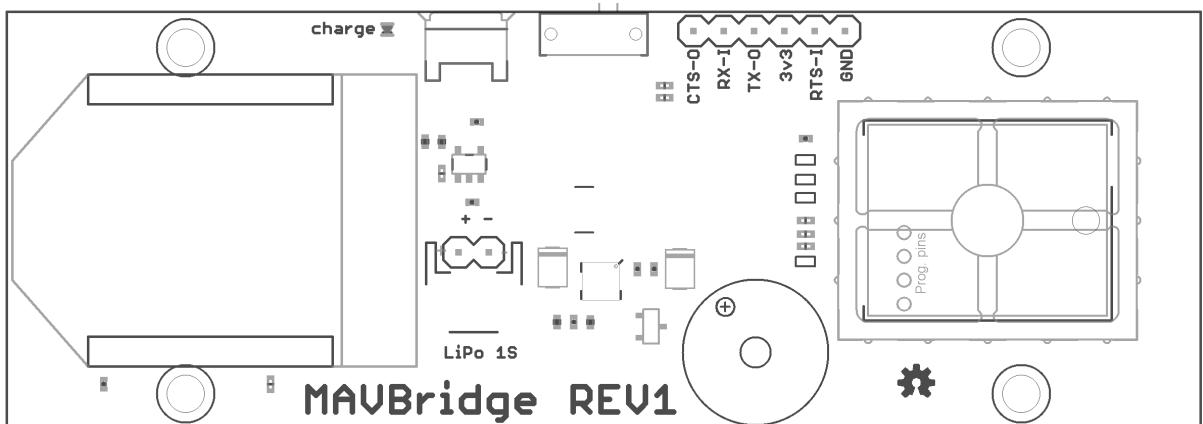
O layout do circuito foi projetado de forma a ser montado em cima da bateria, resultando em um produto compacto. A disposição dos componentes é apresentada na figura 18. As dimensões da placa são de 100x35x5 mm. Quatro furos de 3mm foram colocados nas laterais para fixação.

O módulo de telemetria da Hope-RF está no lado direito da placa, protegido dentro de uma blindagem de metal. A placa foi projetada para ser utilizada com esse módulo montado verticalmente, maximizando a cobertura da antena de telemetria.

No lado esquerdo está montado o módulo de WiFi/Bluetooth, dessa forma maximizando a distância entre as antenas dos dois links de comunicação. Este módulo fica em um soquete, facilitando a troca entre WiFi ou Bluetooth.

Na parte superior estão localizados o conector de carga, a chave liga/desliga e o indicador de carga. Por estarem todos no mesmo lado da placa, a instalação dentro de uma caixa de proteção é facilitada. O conector da bateria está na parte central inferior, e a bateria está localizada atrás da placa de circuito impresso.

Figura 18 - MAVBridge – Layout dos componentes



O layout foi implementado em uma Placa de Circuito Impresso (PCI) de duas camadas, de forma a separar os circuitos de ambos os rádios, circuito de carga e fonte chaveada. O plano de terra presente em ambas as camadas da PCI foi projetado formando uma ligação estrela com o ponto central abaixo do regulador da fonte chaveada. O ponto com

alto ruído de chaveamento presente no terminal do indutor L3 foi reduzido ao menor tamanho possível. O *design* segue as diretivas apresentadas por (TEXAS INSTRUMENTS) no *datasheet* de seu componente TPS61201. A blindagem do radio Hope RF foi conectada ao terra por diversas *vias*, garantindo uma boa conexão e redução de emissões eletromagnéticas. A figura 19 apresenta a camada superior de cobre da placa, e a figura 20 a camada inferior.

Figura 19 - MAVBridge – Layout da camada superior

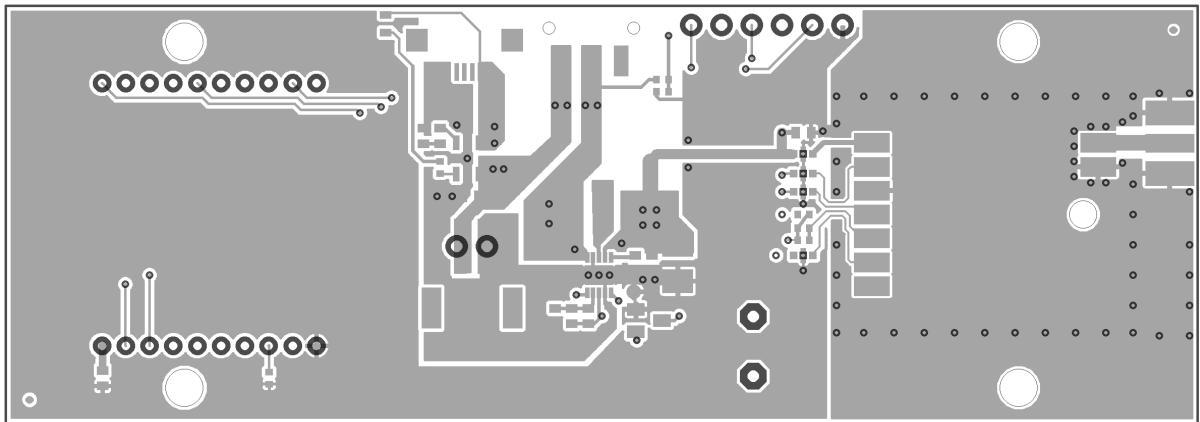
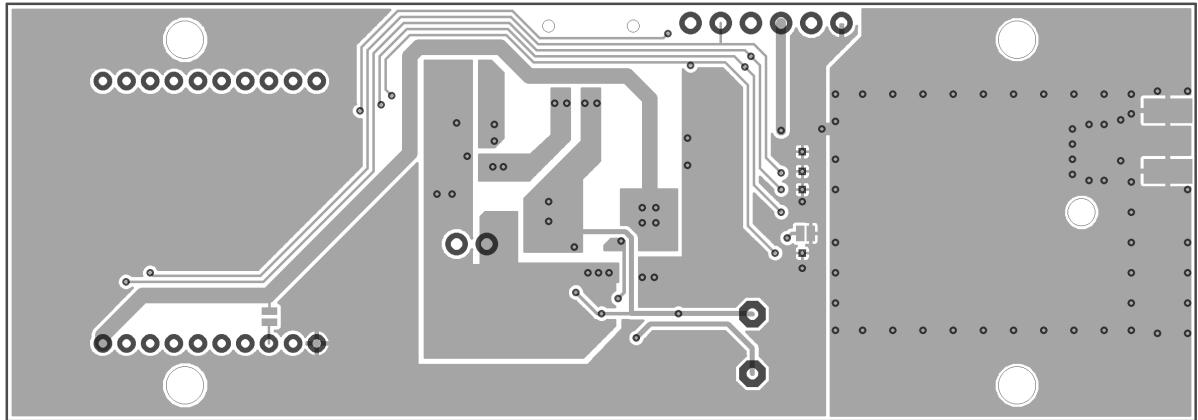


Figura 20 - MAVBridge – Layout da camada inferior



3.3.2 Conexão direta via USB

Como auxiliar ao MAVBridge o *software* também oferece suporte à conexão direta a um módulo de rádio HopeRF ou Xbee. Como a maioria dos dispositivos Android apresenta somente uma porta USB micro-B, um adaptador é necessário, e é encontrado

comercialmente com o nome de cabo *USB-OTG*. A figura 21 apresenta um *tablet* executando o *software* deste projeto com um módulo de rádio contactado.

Figura 21 - Radio Hope RF conectado via porta USB através de um adaptador.



O rádio utilizado neste projeto a fim de testar a conexão direta foi o módulo de telemetria da empresa 3DRobotics. O módulo já inclui o *hardware* com o suporte para conexão USB, e foi projetado visando reduzir os efeitos de emissão eletromagnética (aumentando, dessa forma, a sensibilidade do rádio). A figura 22 apresenta um par de módulos de telemetria com o cabamento para conexão com o *tablet* e VANT.

Figura 22 - módulo de telemetria 3DRobotics



FONTE: 3DRobotics, 2013

4 SOFTWARE

Conforme indicado na revisão de conceitos, o *software* da estação de controle é um dos elementos mais críticos deste sistema (KANG;YUAN, 2009). Este é o foco principal do trabalho, no qual iniciamos o desenvolvimento de um *software* chamado DroidPlanner.

O sistema operacional escolhido foi o Android, devido as vantagens descritas no capítulo de seleção de *hardware*. Dentre as diversas linguagens de programação disponíveis para a programação de aplicativos Android (Java, C, C++, Scala, Groovy, entre outras) a escolhida foi Java, devido a sua grande difusão e ao fato de ser a linguagem principal deste sistema operacional. O *Integrated Development Environment* (IDE) utilizado foi o projeto Eclipse (ECLIPSE), em conjunto com o *Android Software Development Kit* (ANDROID SDK) disponibilizado pelo projeto Android (ANDROID).

O projeto foi desenvolvido utilizando a licença de código aberto GNU GPLv3 (FREE SOFTWARE FUNDATION), o que possibilitou a entrada de novos desenvolvedores no projeto, dado que essa licença garante que o trabalho de todos no projeto continuará livre.

Para facilitar e organizar o desenvolvimento, realizado por diversos desenvolvedores trabalhando em múltiplos ramos do código fonte, o sistema de controle de versão GIT (GIT) foi utilizado. E para disponibilizar o código fonte de forma simples o web-site GITHUB (GITHUB) foi escolhido. Uma das vantagens que este sistema inclui é a facilidade de controlar o trabalho de diversos programadores, tornando o processo de integrar o trabalho de todos em uma única fonte “oficial” simplificada e de rápido acesso, além de possuir um sistema de rastreamento de *program issues* (problemas de funcionamento, solicitações de novos recursos, etc.) e um sistema de *wiki* (tipo específico de coleção de documentos em hipertexto e o *software* colaborativo para criá-lo).

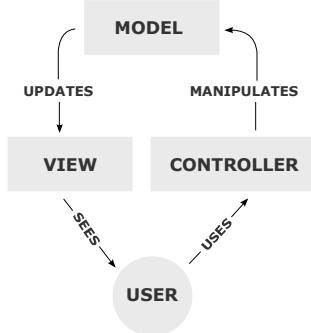
Essas ferramentas levam a um programa mais seguro, testado e revisado por diversas pessoas em diversas condições de trabalho, o que não seria possível caso existisse apenas um desenvolvedor.

4.1 ARQUITETURA

A arquitetura do *software* é baseada no modelo *Model-View-Controller* (MVC), um modelo de arquitetura de *software* que separa três blocos: a informação armazenada, a

representação da informação e a interação do usuário. As ideias centrais por trás do MVC são a reusabilidade de código e separação de conceitos (MCV, Wikipédia). A figura 23 ilustra o conceito com um diagrama de blocos.

Figura 23 - Interação típica dos componentes do modelo MVC



Fonte: MVC, Wikipédia

No caso deste *software* o modelo (*model*) consiste no que chamamos de “VANT virtual”, que contém todos os dados obtidos até determinado momento a respeito da aeronave. O modelo gera notificações aos outros componentes quando há uma mudança em seu estado.

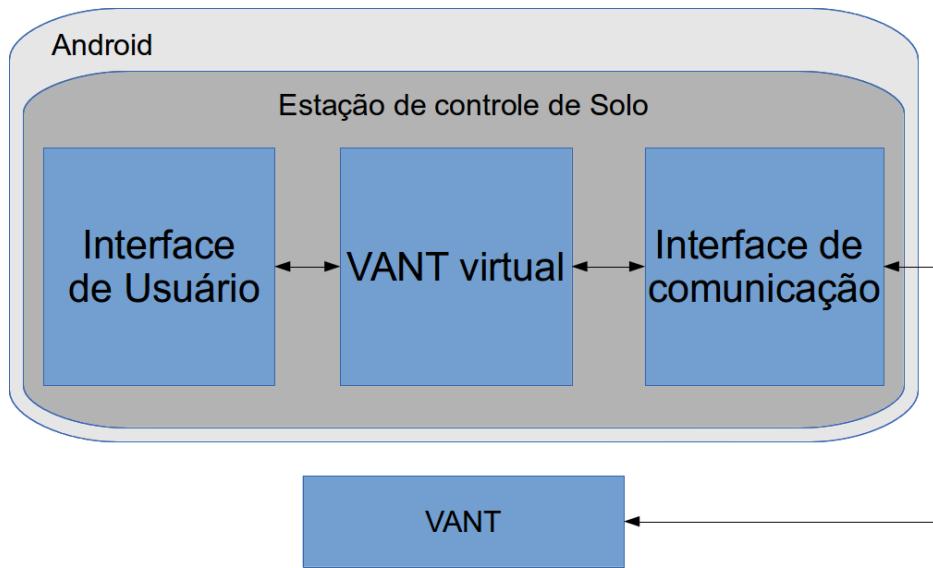
A o bloco *view* representa as diversas telas de interface de usuário, que são a saída de representação dos dados como o *Heads Up Display* (HUD) e o mapa. Estes componentes são independentes entre si, sendo atualizados quando recebem notificações do modelo. As interfaces de usuário também podem atuar no “VANT real” ao enviar comandos de usuário ao controlador.

O controlador (*controller*) recebe todo o tráfego de telemetria, convertendo-o em comandos para modificar o modelo (que notifica os componentes de visão). Os dados também trefegam no outro sentido, isto é, quando uma interface de usuário é acionada, o modelo é modificado e os comandos necessários são enviados ao “VANT real”. Nessa implementação o controlador está ligado diretamente ao “VANT virtual”, garantindo que as transações de dados (como uma missão autônoma) sejam sincronizadas entre o modelo e a aeronave “real”.

Adicionalmente aos componentes do modelo MVC foi desenvolvido uma interface de comunicação, devido aos diversos tipos de protocolos utilizados (USB, Bluetooth, TCP, UDP) para comunicação com o “mundo externo”. Esta interface é abstrata, sendo implementada por objetos que administram cada um dos tipos de comunicação.

A figura 24 apresenta um diagrama de blocos geral do sistema implementado. O aplicativo é executado sobre o sistema operacional Android, com os seus três componentes principais: Interface de usuário, VANT virtual e a interface de comunicação, comunicando-se com a aeronave real pelo link de telemetria.

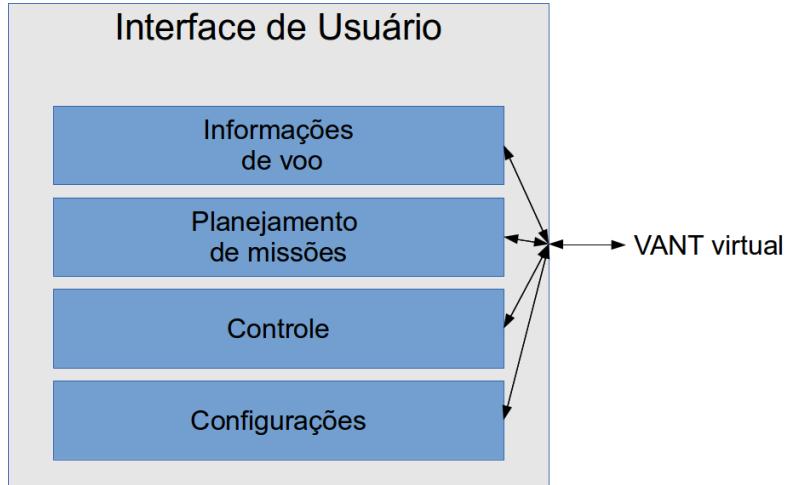
Figura 24 - Diagrama de blocos da arquitetura do software



4.1.1 Interface de usuário

O objetivo da interface de usuário é de apresentar as informações e possibilitar a interação com a aeronave. Ela é o meio de comunicação do usuário com o *software*, e deve realizar essa função de forma simples. A comunicação destes componentes é realizada através do VANT virtual, abstraindo todo o conceito de protocolos de comunicação. Para limitar o número de informações as quais o usuário tem de processar, a interface foi dividida em diversas telas, cada uma utilizada em uma parte da operação da aeronave. A figura 25 apresenta um diagrama exemplificando esses componentes.

Figura 25 - Diagrama de blocos da interface de usuário



A tela principal é a de informações de voo, apresentando dados sobre a operação da aeronave. O seu objetivo é informar o usuário sobre o estado atual do VANT, de forma rápida e clara. Pequenos comandos podem ser realizados por essa interface, como a mudança de modo de voo, por exemplo.

A interface de planejamento serve para criar e editar missões autônomas, e, normalmente, é utilizada na preparação de uma missão para o VANT. Como uma missão tem diversos detalhes (posicionamento dos *waypoints*, tipo de *waypoints*, ordenamento da missão, entre outros) os dados são exibidos de diversas formas, como: mapas, listagens e diálogos de detalhes. Outra função é a de criação de missões para objetivos de aerofotogrametria, que possui diversas características próprias que serão exploradas em um capítulo posterior.

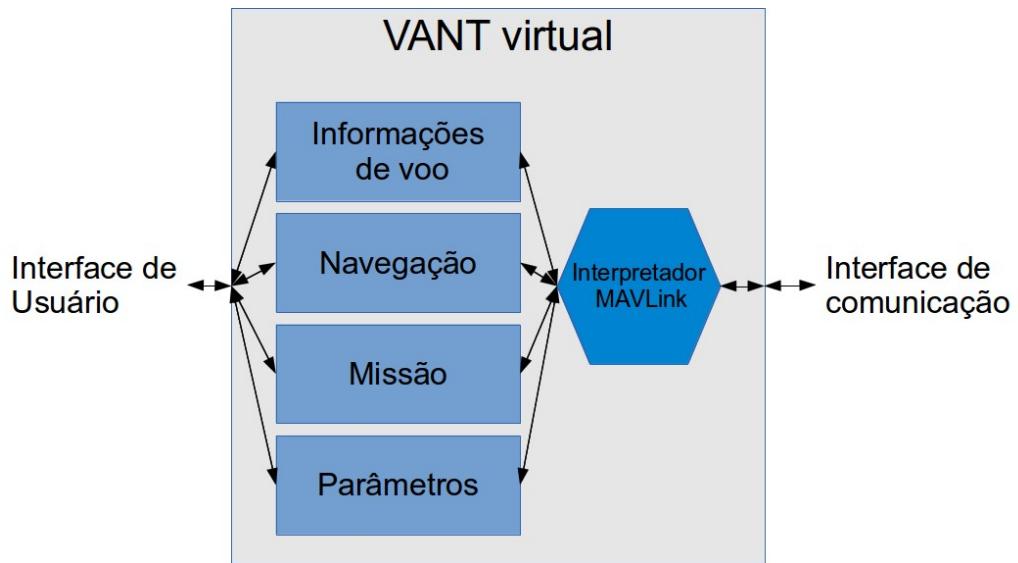
O controle direto da aeronave tem a sua própria interface, pois caso a aeronave esteja sendo operada dessa maneira as necessidades do usuário mudam. Nesse caso o foco do usuário muda de “acompanhar a missão” para “controlar a aeronave”. Como são utilizados controles virtuais na tela do *tablet* o espaço disponível também fica limitado.

Como os VANTS são veículos complexos eles normalmente apresentam diversos parâmetros, os quais devem ser configurados para uma operação correta. O mesmo é valido para o *software* da estação de controle, que contém configurações como o tipo de meio de comunicação a ser utilizado. A visualização e configuração desses parâmetros é o propósito da tela de configurações.

4.1.2 VANT virtual

O VANT virtual é uma mistura de dois componentes do modelo MVC, o qual inclui o modelo e o controlador. O modelo é formado por uma estrutura de dados contendo as informações recebidas até o momento sobre a aeronave real, e os dados são armazenados em objetos internos divididos em categorias. A figura 26 mostra um diagrama de blocos desse componente onde deve-se observar que esse é um diagrama simplificado, não apresentando todas as categorias de armazenamento de dados utilizadas.

Figura 26 - Diagrama de blocos do VANT virtual



As informações são geradas pelo controlador que, nesse caso, é o interpretador MAVLink, que decodifica mensagens do protocolo MAVLink atualizando o estado do modelo. Sempre que uma atualização é realizada, notificações para as interfaces de usuário são geradas, atualizando as informações apresentadas ao usuário. Exemplificando: uma mensagem de posicionamento GPS é transmitida do VANT via telemetria, o controlador recebe a mensagem, a decodifica e atualiza a posição do modelo, gerando uma notificação e atualizando o marcador de posição da aeronave na interface de mapa. Assim, não é necessário realizar a comunicação com o veículo sempre que uma informação for requisitada pela interface de usuário, pois a cópia local é utilizada.

A interface de usuário também interage com esse VANT virtual através de comandos, enviados para o VANT real e que, após confirmação, atualizam esse objeto. Esse

processo de comunicação reduz o tráfego através do link de telemetria, que possui uma taxa limitada. Um exemplo típico é o envio de uma nova missão a partir da interface de usuário, cujo processo inicia-se com a geração da missão pela interface de usuário, que é enviada para o VANT virtual, onde o interpretador MAVLink realiza a transação da missão com a aeronave e, quando concluída com sucesso, a missão do modelo é atualizada.

As informações do modelo são separadas em categorias. Cada categoria é representada por uma classe de objetos no programa. Abaixo são listadas as categorias utilizadas para armazenamento de dados, com alguns exemplos de dados típicos:

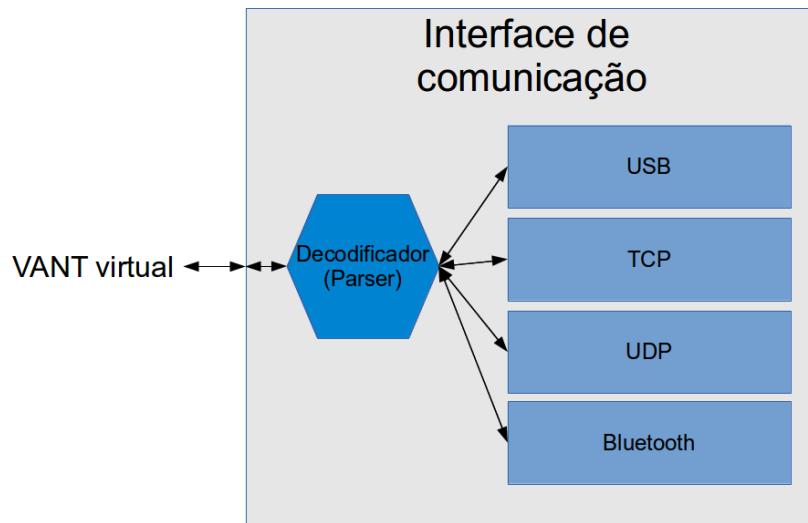
- Tipo do VANT conectado à estação de controle (asa-fixa, multimotor, outros)
- Estado do sinal de GPS
- Missão autônoma programada
- Estado do rádio controle de *backup*
- Velocidade do VANT nos diversos referenciais
- Estado geral da aeronave (voando, armada, em modo de segurança)
- Estado da bateria (tensão, corrente atual, percentual de carga)
- Localização do ponto de retorno em caso de *failsafe* (minimização de risco em caso de falhas)
- Estatísticas da missão autônoma (percentual concluído, próximo *waypoint*)
- Altitude da aeronave
- Orientação espacial (ângulos de Euler, quaterniões)
- Estado do controlador de navegação
- Parâmetros de configuração da aeronave

Sendo as categorias objetos da linguagem Java, o comportamento de cada uma pode ser controlado, pois a estrutura de dados utilizada para armazenamento não é exposta ao resto do programa. Tomando o estado do GPS como exemplo, diversos métodos podem ser utilizados para obter informações como: “A posição é valida?”, “Qual a última posição?”, “Qual o estado do GPS?”, “Qual o número de satélites dos quais o sinal está sendo recebido?”. Estas informações são utilizadas pela interface de usuário, enquanto a estrutura de dados interna a esse objeto fica protegida de acesso externo.

4.1.3 Meios de Comunicação

O último componente do sistema é a interface de comunicação, cujo objetivo é realizar a comunicação do *software* com o link de telemetria. Como os dados nos quais o programa tem interesse são transmitidos pelo protocolo MAVLink e os meios de transmissão transmitem dados de forma binária, foi necessária a implementação de um decodificador de pacotes MAVLink (*parser*). A figura 27 apresenta o componente de comunicação em forma de um diagrama de blocos.

Figura 27 - Diagrama de blocos da interface de comunicação



Conforme explicitado no subcapítulo 3.3, diversos tipos de comunicação podem ser requisitados da estação de controle. Para resolver esses problemas a interface foi implementada como um objeto abstrato, o qual contem métodos de escrita e leitura. Cada tipo de protocolo de comunicação utilizado tem uma classe implementando esse objeto abstrato, com os seus próprios métodos específicos de entrada e saída. Os meios de comunicação que foram implementados (e o *hardware* com o qual eles são utilizados) estão listados abaixo:

- USB – Comunicação direta, módulo HopeRF ou Xbee
- Bluetooth – MAVBridge ou comunicação direta a curta distância
- TCP – Link de comunicação 3G ou via MAVBridge
- UDP – Link de comunicação WiFi ou MAVBridge

O decodificador (*parser*) tem como objetivo transformar o fluxo de dados, recebido via um dos meios de comunicação, em objetos representando mensagens do protocolo

MAVLink. Isso é feito detectando um dos indicadores de início de pacote, detecção do tipo de mensagem, coleta dos dados da mensagem e verificação de integridade de pacote. Mais informações sobre o protocolo MAVLink estão descritas no capítulo de revisão de conceitos.

Quando uma mensagem é recebida pelo decodificador, ela é enviada para o controlador no VANT virtual, atualizando indiretamente o modelo e a interface de usuário. O mesmo ocorre no sentido inverso, isto é, uma mensagem enviada pelo VANT virtual é codificada no formato do protocolo MAVLink e transmitida por um dos meios de comunicação.

4.2 INTERFACE HOMEM MÁQUINA

4.2.1 Informações de voo

A tela de informações de voo tem como função criar uma percepção da situação da aeronave no espaço. A figura 28 contém a captura dessa tela, onde no lado esquerdo temos o *Heads Up Display* (HUD), detalhado posteriormente, e à direita o mapa da situação.

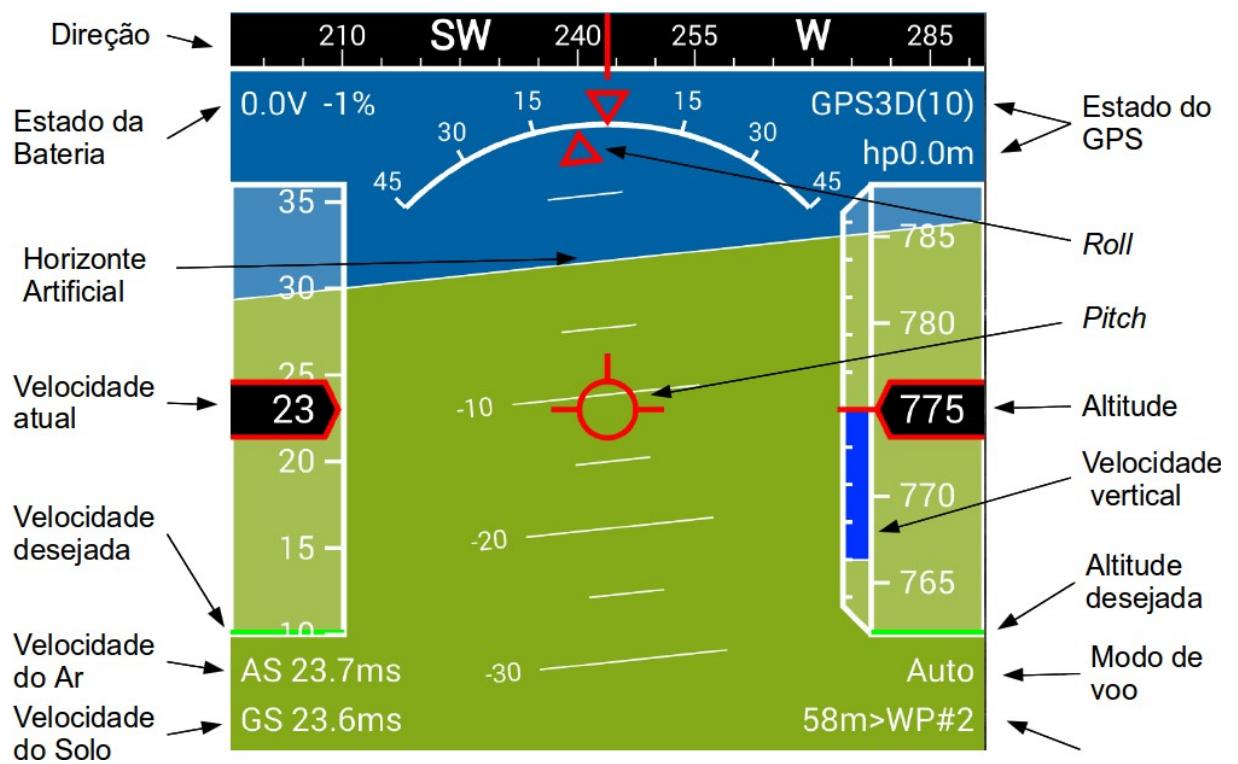
Figura 28 - Tela de Informações de voo



O mapa apresenta a missão que está programada no VANT, indicada pelos marcadores 1 a 5 (cada um apresentando um *waypoint*, com a sua altitude). A aeronave (neste caso um avião asa-fixa) é representada como um marcador vermelho próximo ao *waypoint* 2. As últimas coordenadas da aeronave são representadas por uma trilha azul. Com o auxílio dos menus superiores é possível realizar a troca do *waypoint* para o qual a aeronave prosseguirá ao se ativar o modo de missão autônoma. Um segundo menu possibilita a troca do modo de voo.

A figura 29 mostra o HUD detalhando os componentes visuais, no qual o *design* foi baseado em *displays* utilizados em aeronaves tripuladas. A orientação da aeronave é exibida com o auxílio de um horizonte artificial e de marcas indicadoras de ângulo de rolamento (*roll*) e arfagem (*pitch*). O indicador esquerdo exibe a velocidade atual da aeronave (em relação ao solo ou ar, dependendo dos sensores instalados) e a velocidade de referência (velocidade desejada) do controlador de navegação através de uma barra verde. Dois indicadores textuais na parte inferior esquerda mostram as velocidades do ar e de solo. O estado da bateria (tensão, corrente sendo consumida, porcentagem de carga restante) é exibido em um elemento textual na parte superior esquerda, caso o VANT esteja enviando esses dados. O indicador direito exibe a altitude atual (indicado com contorno vermelho), a altitude desejada (utilizando uma barra verde), e a velocidade vertical (barra azul na lateral esquerda do indicador). Informações sobre a missão que será executada (numeração e distância do próximo *waypoint* da missão, modo de voo atual) são exibidas abaixo do indicador de altitude. Acima deste indicador é exibido o status do sistema de GPS, como o tipo de coordenadas obtido, número de satélites e diluição horizontal da coordenada obtida. O eixo de guinada é exibido na parte superior.

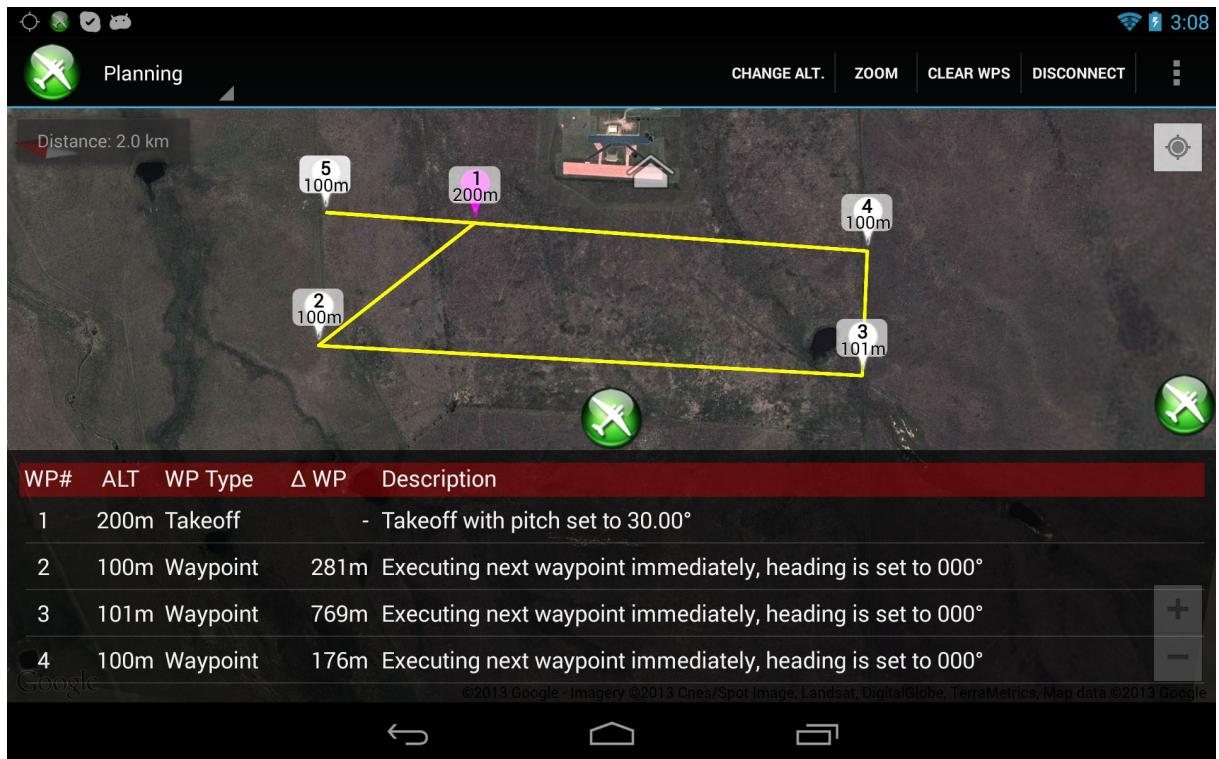
Figura 29 - Detalhes do Heads Up Display



4.2.2 Planejamento de missões

O planejamento de missões autônomas é realizado com o auxílio da tela de planejamento capturada na figura 30. Nesta tela os *waypoints* são adicionados ao realizar um clique longo sobre a coordenada desejada no mapa. Como forma alternativa de adição de *waypoints* foi implementado um modo de detecção de trajetória no qual o usuário desenha a rota desejada sobre a tela *touch-screen* e os pontos da trajetória são coletados pelo *software* e posteriormente processados utilizando um algorítimo de simplificação de linhas gerando um conjunto de *waypoints*.

Figura 30 - Tela de Planejamento de missões



O menu inferior apresenta uma listagem de todos os *waypoints*, com a sua respectiva altitude, tipo, distância ao *waypoint* anterior e descrição do comando que será executado por esse *waypoint*. A listagem também possibilita a exclusão de *waypoints*, através de um gesto de deslizamento horizontal sobre o mesmo. Outra interação com a listagem é feita através de um gesto de arrastar, de maneira a realocar o *waypoint*. O menu oferece opções para enviar a missão para o VANT, receber a missão atual da aeronave e salvar e ler missões de um arquivo.

Ao selecionar um *waypoint* tocando na listagem ou no mapa, a janela capturada na figura 31 é exibida, onde são apresentadas informações extras sobre o *waypoint*, e possibilita a configuração de parâmetros como altitude, raio do *waypoint* e *delay* aplicado a este *waypoint*, além de uma descrição do funcionamento deste *waypoint*. Foi projetada uma janela para cada tipo de *waypoint* disponível no conjunto básico do protocolo MAVLink (*Waypoint*, *Loiter*, *Region of Interest*, *Return To Land*, *Takeoff*, *Land*, *Path*).

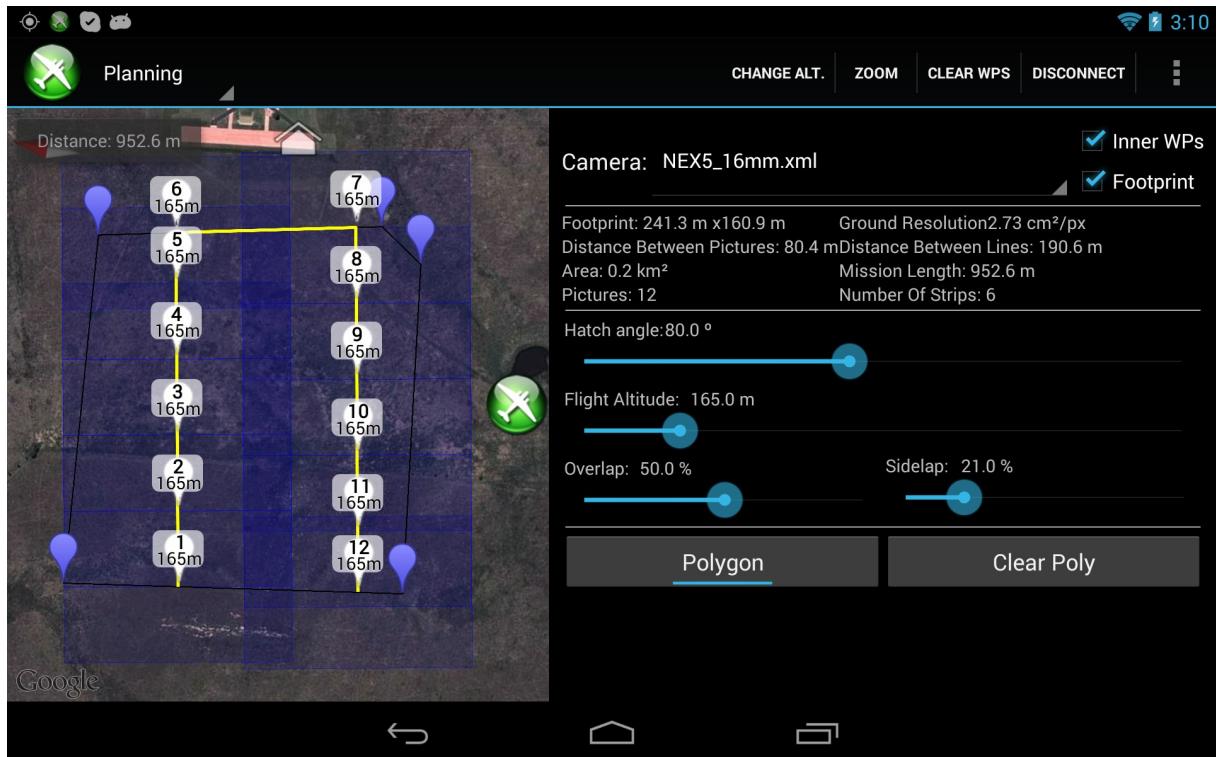
Figura 31 - Tela de Planejamento de missões



Para auxiliar no desenvolvimento de missões para objetivos de aerofotogrametria, o “diálogo” (janela de interação com o operador) exibido na figura 32 foi criado. O princípio de funcionamento dessa janela inicia-se com o desenho de um polígono sobre a área que se deseja mapear. A câmera que será utilizada é selecionada no componente superior (um conjunto de câmeras tipicamente usadas em VANTs de pequeno porte já foi programada no *software*, com a possibilidade do usuário adicionar arquivos de câmeras personalizados). Os parâmetros do mapeamento (altitude do voo, sobreposição lateral e longitudinal das fotos, ângulo com a vertical das linhas de voo) podem ser modificados e a missão é dinamicamente atualizada no mapa, e algumas informações sobre a missão são

exibidas no topo da janela (distância total da missão, resolução, número e tamanho projetado das imagens, tamanho da área mapeada, distância longitudinal e lateral entre as fotos).

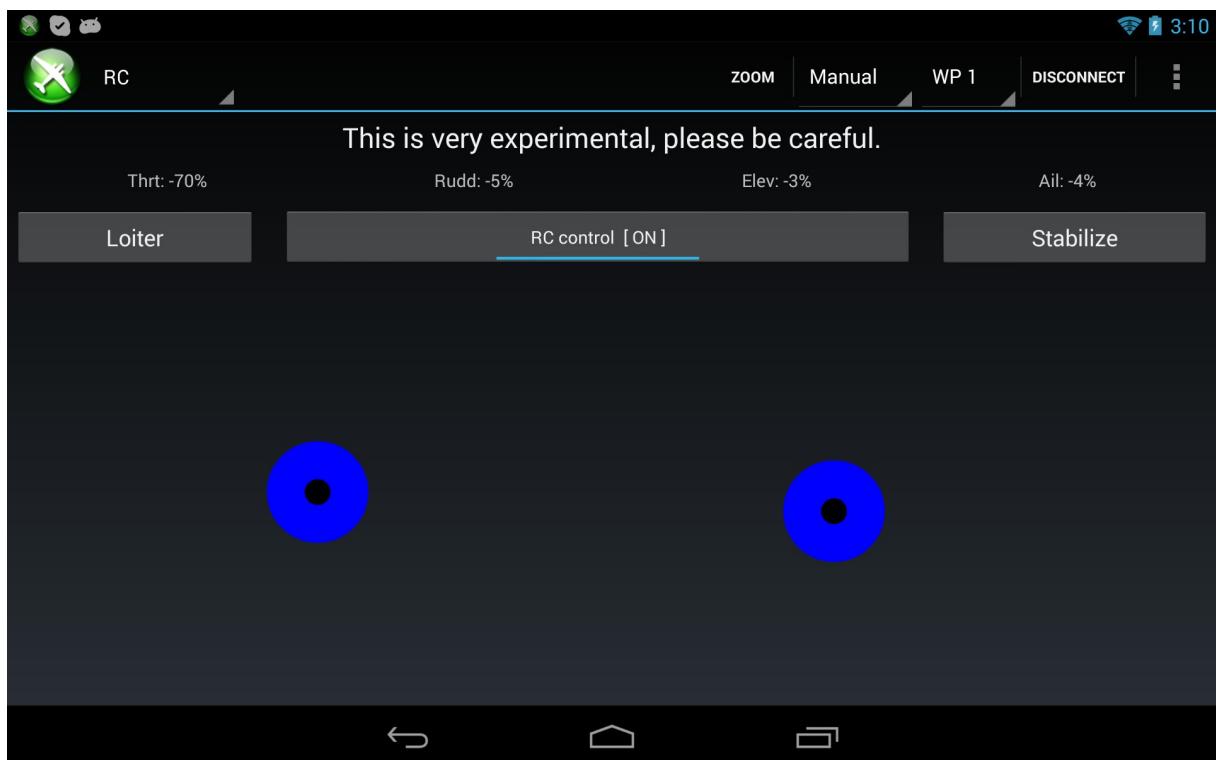
Figura 32 - Tela de Planejamento aerofotogramétrico



4.2.3 Controle

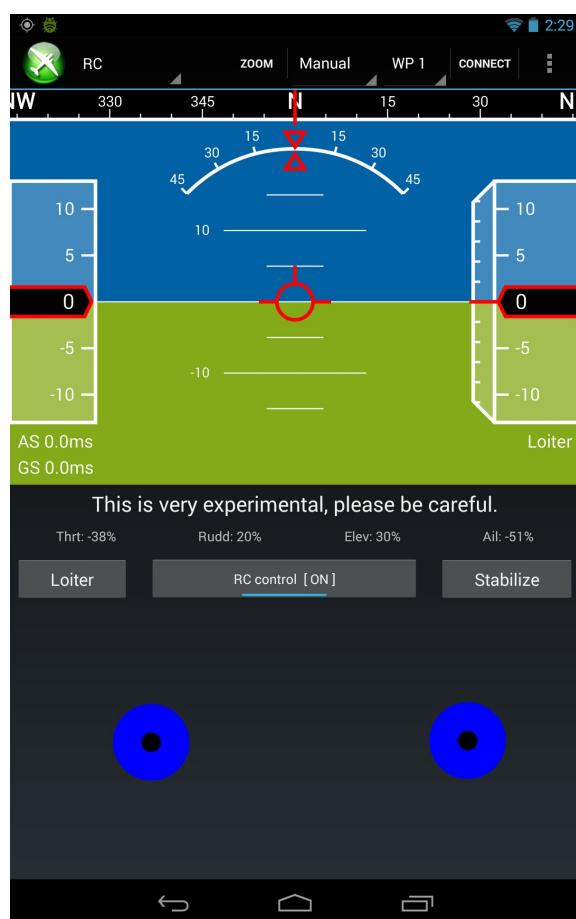
Para o controle direto da aeronave a tela capturada na figura 33 é utilizada. Dois *joysticks* virtuais são exibidos e, quando o usuário toca a tela, o movimento relativo a partir do ponto de toque é utilizado como deflexão do *joystick*, e esse dado é transmitido ao VANT. Um botão “interruptor” ativa ou desativa o controle por essa interface. Dois botões auxiliares possibilitam a troca rápida de modo de voo.

Figura 33 - Tela de Controle – visualização horizontal (paisagem)



No caso da utilização desta tela na orientação vertical o *layout* apresentado na figura 34 é utilizado. Contendo uma mistura das informações contidas no *HUD* (metade superior) e a tela de controle (metade inferior).

Figura 34 - Tela de Controle – visualização vertical (retrato)



4.2.4 Parâmetros

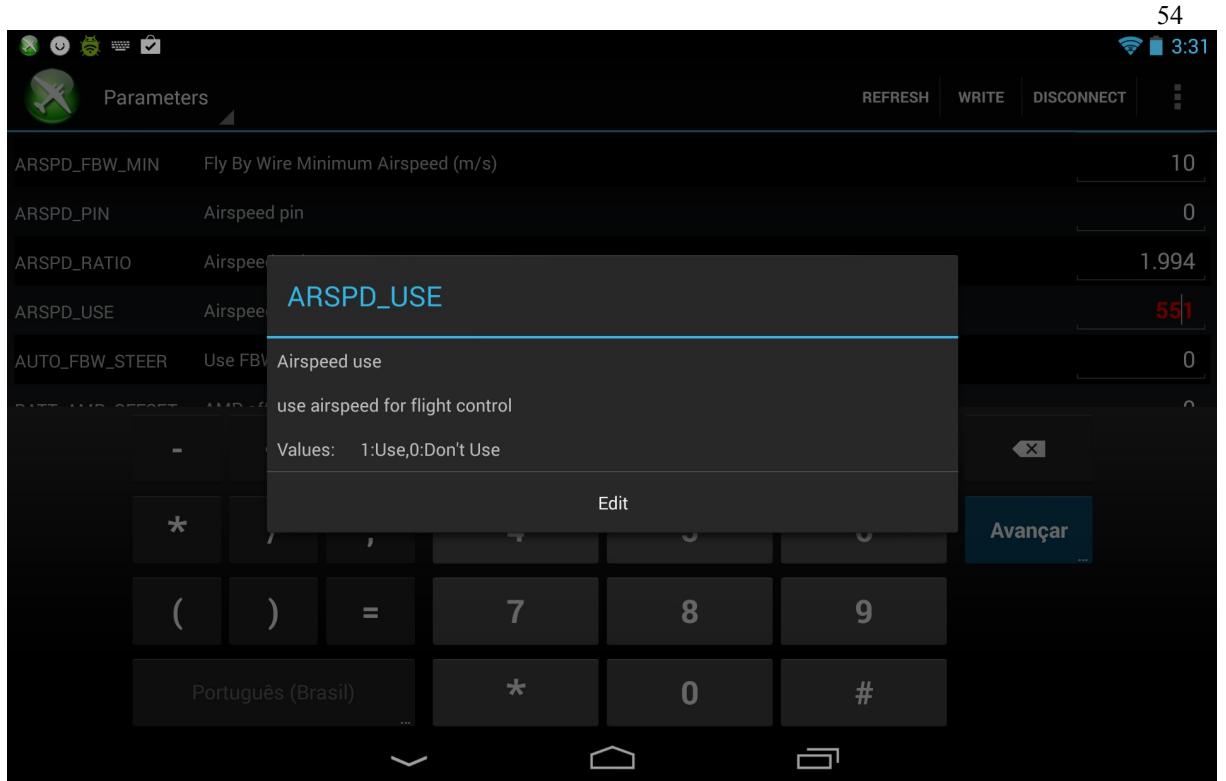
A configuração dos parâmetros do VANT é realizada com a interface de parâmetros, apresentada na figura 35. Os parâmetros são listados com nome, descrição simplificada e valor (essas informações estão contidas nas definições do protocolo MAVLink). Os valores podem ser alterados com o teclado virtual do sistema Android, e caso alterados para um valor diferente do configurado no veículo a fonte é modificada para a cor vermelha.

Figura 35 - Tela de Parâmetros



Caso uma das descrições seja selecionada a tela apresentada na figura 36 é exibida, contendo uma descrição mais detalhada sobre o parâmetro e valores típicos, máximos e mínimos (caso aplicável).

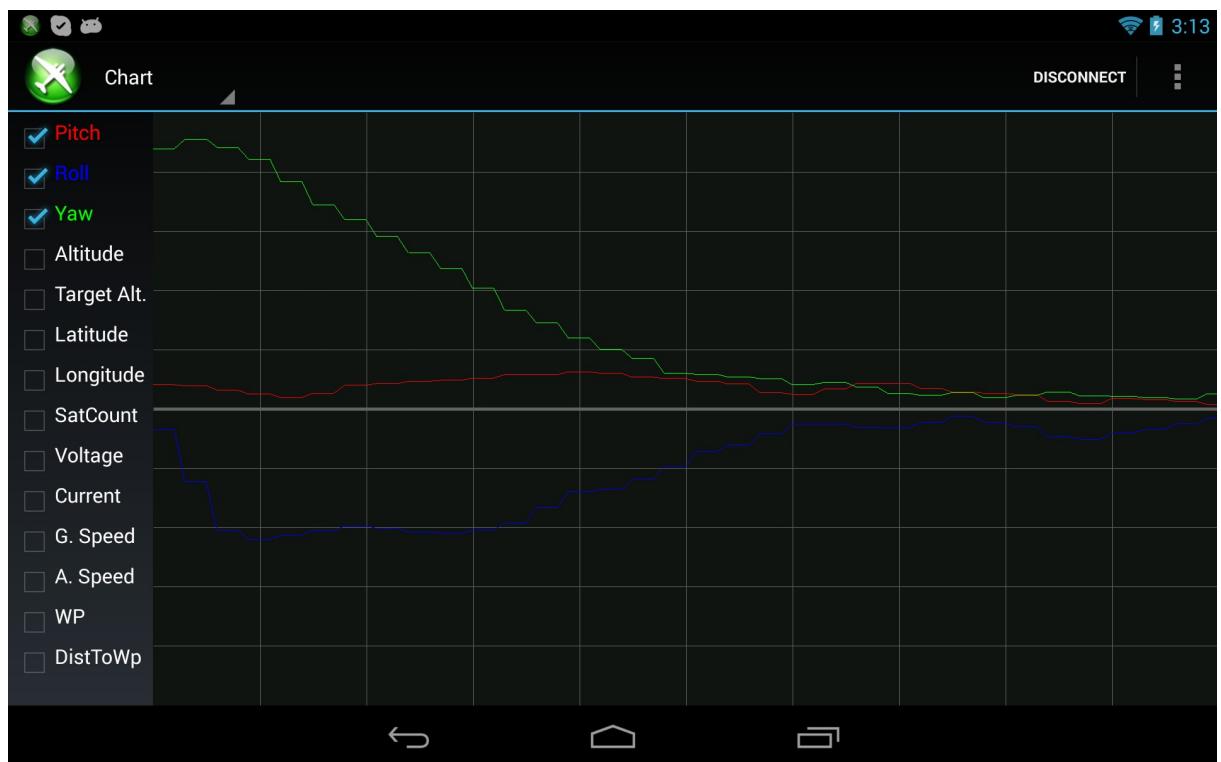
Figura 36 - Exemplo de tela de detalhe de parâmetros



4.2.5 Gráficos

Para facilitar o ajuste dos sistemas de controle PID, a interface de visualização gráfica é utilizada, plotando variáveis do sistema (eixo vertical) *versus* o tempo (eixo horizontal). As variáveis a serem exibidas são selecionadas na listagem a esquerda, e as cores são reutilizadas através de um *stack* de cores (pilha). A captura dessa tela é apresentada na figura 37.

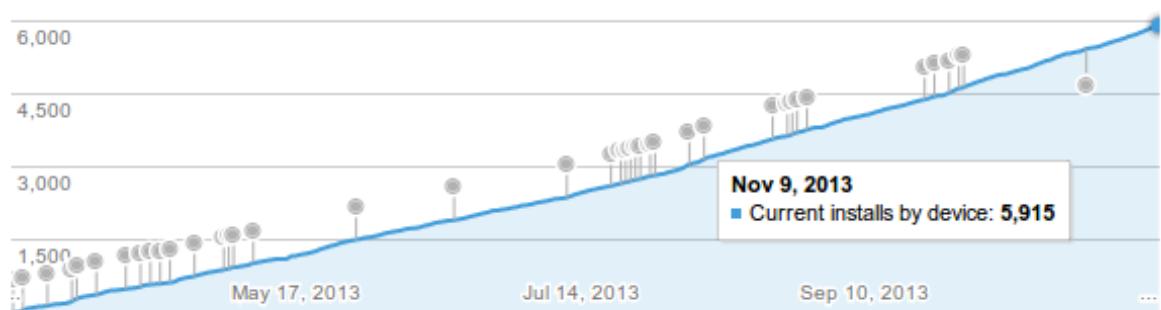
Figura 37 - Tela de Gráficos



5 RESULTADOS

Dado que o projeto foi desenvolvido sob uma perspectiva *open-source*, houve a colaboração de diversos desenvolvedores, e como foi disponibilizado com fácil acesso através do website Google Play (GOOGLE – DroidPlanner) um número ainda maior de usuários foi obtido. A figura 38 mostra o crescimento do número de usuários com o aplicativo instalado a partir da data de publicação do projeto, chegando a um total de 5915 usuários na data de 9 de novembro de 2013.

Figura 38 - número de dispositivos com o aplicativo instalado



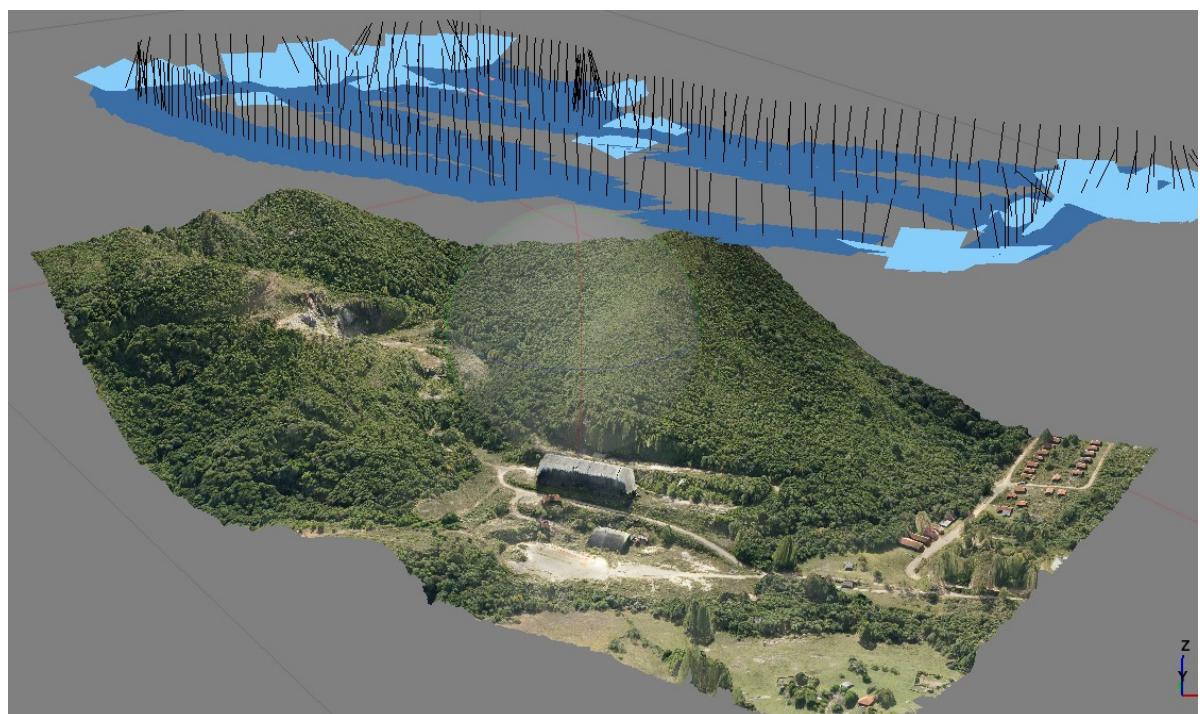
Por ter sido utilizado um sistema de controle de versão (GIT) é possível obter estatísticas sobre a colaboração dos desenvolvedores no decorrer deste projeto. Alguns destes dados são apresentados na listagem abaixo (mais informações podem ser obtidas pelo website GITHUB):

- 22 desenvolvedores
- 1226 sub-versões (*commits*)
- Traduzido para 12 línguas diferentes
- ~ 50000 linhas de código fonte
- 970 classes/arquivos de código fonte
- 10 meses de desenvolvimento

Combinando os conceitos apresentados anteriormente acerca de fotogrametria com o desenvolvimento deste projeto, foi realizado um voo experimental com o objetivo de mapeamento de uma região. A rota de voo foi planejada com a interface de aerofotogrametria

apresentada no capítulo 4.2.2, utilizando os parâmetros de mapeamento com valores comerciais. O voo foi executado de forma autônoma, com o acompanhamento realizado pela interface de informações de voo. Os resultados foram satisfatórios. A figura 39 apresenta o modelo topográfico do terreno (os retângulos azuis são os locais onde as fotos foram coletadas pelo VANT).

Figura 39 - Exemplo de resultados fotogramétricos



6 PROPOSTAS DE MELHORIAS

7 CONCLUSÕES

Referências

BEARD, R. W.; McLAIN, T. W. Small Unmanned Aircraft. 1st ed. Princeton University Press, 2012.

KANG, W.; YUAN, M. Software Design for Mini-type Ground Control Station of UAV. School of Automation Science and Electrical Engineering, Beijing University of Aeronautics & Astronautics, 2009

PIGNATON, E. Cooperative Context-Aware Setup and Performance of Surveillance Missions Using Static and Mobile Wireless Sensor Networks. PhD thesis, 2011.

FONTANARI, Athos Alexandre Lima; RECH, Flávio; PEREIRA, Carlos Eduardo. Sistema de Planejamento e Controle de Missão de Um Véículo Aéreo Não-Tripulado. 2011.

PEREIRA, C. E. Integração de redes de sensores sem fio com veículos aéreos não-tripulados (VANTS). Premio Santander 2010, 2010.

HOPE MICROELECTRONICS CO., LTD. HM-TRP Series 100mW Transceiver modules V1.0., 2006:Datasheet.

3DROBOTICS. Disponível em: <<http://www.3drobotics.com>>. Acesso em: 17 outubro 2013.

SKYDRONES. Disponível em: <<http://www.skydrones.com.br>>. Acesso em: 2 novembro 2013.

DROIDPLANNER. Disponível em: <<https://github.com/arthurbenemann/droidplanner>>

DROIDPLANNER – COMPATIBLE DEVICES. Disponível em:
<<https://github.com/arthurbenemann/droidplanner/wiki/Compatible-Devices>>

SIK. Disponível em: <<https://github.com/tridge/SiK>>

FREE SOFTWARE FUNDATION. Disponível em: <<https://gnu.org/licenses/gpl.html>>

ECLIPSE. Disponível em: <<http://www.eclipse.org/>>

ANDROID. Disponível em: <<http://source.android.com/>>

ANDROID SDK. Disponível em: <<http://developer.android.com/sdk/index.html>>

GIT. Disponível em: <<http://git-scm.com/>>

GITHUB. Disponível em: <<https://github.com/>>

WIKIPEDIA – MCV. Disponível em: <<http://en.wikipedia.org/wiki/Model-view-controller>>

GOOGLEPLAY – DROIDPLANNER. Disponível em:
<<https://play.google.com/store/apps/details?id=com.droidplanner>>

APÊNDICE A – Esquemático MAVBridge

