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

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Engenheiro Elétrico

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

2013

AGRADECIMENTOS

A 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 seu auxílio nas tarefas desenvolvidas no desenvolvimento e revisão deste trabalho.

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 eletrical 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.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

UAV: *Unmanned Aircraft Vehicle*

VANT: *Veiculo Aereo Não Tripulado*

GCS: *Ground Control Station*

MAVLink: *Micro Air Vehicle Link*

ASCII: *American Standard Code for Information Interchange*

CRC: *Cyclic Redundancy Check*

LSB: *Least Significant Bits*

MSB: *Most Significant Bits*

XML: *Extensible Markup Languages*

UVLO: *UnderVoltage LockOut*

LiPo: *Lithium Polymer Battery*

LASCAR: Laboratório de Sistemas de Controle, Automação e Robótica

VARP: Veículo Aéreo Remotamente Pilotado

IHM: Interface Homem Maquina

2 INTRODUÇÃO

Nos últimos anos, 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.

3 REVISÃO DE CONCEITOS

3.1 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 veiculo 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.

1. Plataforma ArduCopter

FIGURA 1. 3DROBOTICS ARDUCOPTER Y6

Fonte: 3DRobotics, 2013

3.2 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.

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. 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 autonoma pelo VANT).

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).

3.3 REDES SEM FIO

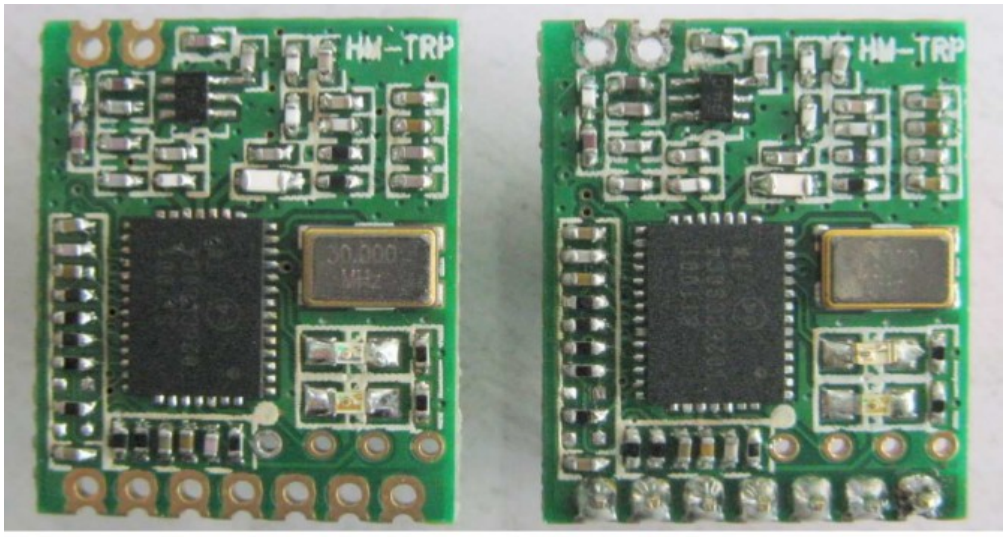
1. XBEE

Figura 2. MODULO XBEE-XSC

FONTE: DIGI, 2013

2. HOPE-RF

FIGURA 3. MODULO HM-TRP



FONTE: HOPE MICROELECTRONICS, 2013

3. BLUETOOTH

FIGURA 4. MODULO RN-42

FONTE: MICROCHIP, 2013

4. WIFI

Figura 5. Modulo RN-171

FONTE: MICROCHIP, 2013

3.4 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, 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, com sintaxe similiar às linguagens C/C++. Para o projeto descrito neste documento, a linguagem de programação utilizada foi Java.

Figura 6. Logo do sistema operacional Android

3.5 PROTOCOLO DE COMUNICAÇÃO

O protocolo de comunicação utilizado foi o MAVLink v1.0. Este foi escolhido pois existem 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 [], e liberado para uso sobre a licença LGPL em 2009. O uso mais comum e 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 contém uma estrutura parecida com o protocolo CAN. Cada pacote contém uma mensagem de tamanho variável, cujos os dados depende 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 uma certa flexibilidade sobre o custo de se tornar incompatível com as GCS disponíveis.

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 sequencia
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	(n+7) a (n+8)	Check-sum do pacote, para detecção de erros de transmissão

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, caso ocorra uma diferença a mensagem é descartada.

Outra função do campo de CRC é garantir que o transmissor e receptor concordam com o tipo de dado sendo transferido. Isso é feito adicionando uma semente ao gerador de checksum baseada em um 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.

3. Mensagems

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 tipica do protocolo, extraida 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 sytem, 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>
```

4. Transferência de missões

A transmissão de uma missão é realizada através de diversas transações, cada transação é validada podendo ser requerido pelo receptor uma retransmissão. As transferências podem ser realizadas no sentido GCS para VANT, e vice-versa.

4 HARDWARE

O objetivo deste projeto é o desenvolvimento de uma estação de controle de solo para VANTs, com o foco principalmente em pequenos veículos. Normalmente estes veículos são operados em linha de visão, com pouco tempo para configurar uma base de comando, por isso nos limitamos 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 o dividimos em estação de controle, VANT e link de telemetria. Cada item sera detalhado nos capítulos seguintes.

4.1 ESTAÇÃO DE CONTROLE

A estação de controle é o equipamento que sera utilizado em campo para o controle, visualização, e planejamento do voo do VANT. Para atendermos todos esses itens de projeto, e tentando reduzir o custo do sistema, optamos pela utilização de dispositivos Android disponíveis no mercado. Por serem um produto o qual já esta bem difundido comercialmente, eles apresentam as seguintes vantagens:

- Programação simples
- Altamente disponíveis
- Baixo custo
- Alto poder computacional em relação ao preço
- Baixo peso
- Portatil

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

Figura 7. Nexus 7

Porem como a única restrição deste projeto quanto a estação de controle é que o hardware rode o sistema operacional Android a gama de dispositivos disponíveis é grande. Alguns dos dispositivos no qual 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*).

4.2 VANT

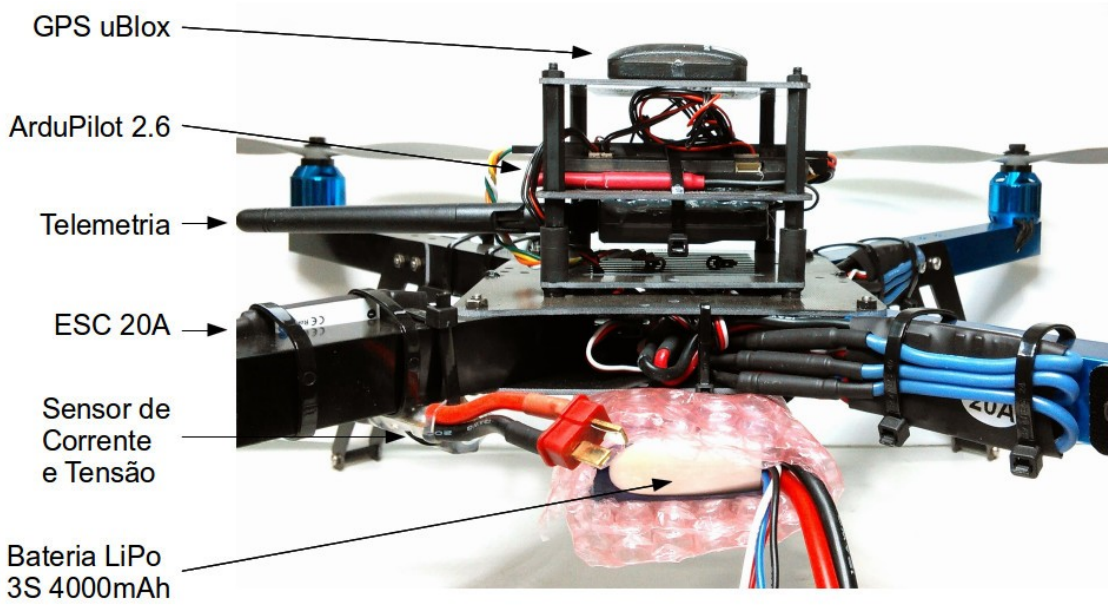
Este projeto suporta o uso de diversos tipos de VANT, como veículos multi-rotores e asa-fixa, porem para possibilitar a realização de experimentos um veiculo do tipo asa fixa foi construído. Um kit de quadcoptero (multi-rotor com 4 motores) da empresa 3DRobotics foi utilizado. O sistema de controle *on-board* é o ArduPilot, o qual é um projeto open-source para controle de VANTs, o qual é compatível com o protocolo MAVLink. Uma foto do protótipo montado é apresentada na figura XX.

Figura 8. 3DR ArduCopter Quad C Frame

Os componentes utilizados na montagem deste quadcopteros estão listados abaixo, e apresentados na figura XX.

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

Figura 9. Componentes do quadcoptero



4.3 LINK DE COMUNICAÇÃO

O link de comunicação é vital para o funcionamento correto da estação de controle, pois por ele que são transmitidos e recebidos todos os dados da aeronave. Como a aplicação deste projeto é para a operação de pequenos VANTs, os nossos 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 de energético, pois estamos limitados a baterias tanto no VANT quanto na Estação de controle
- Pequenas dimensões, novamente estamos limitados a portabilidade da estação de controle, e do tamanho do VANTs.

Como o objetivo do projeto é o desenvolvimento da estação de controle, nos limitamos a procurar soluções comercialmente disponíveis. Os links de radio digitais que foram encontrados são: Xbee, HopeRF, WiFi e Bluetooth (para mais informações consulte o capitulo de revisão de conceitos).

O link de Xbee foi utilizado inicialmente, porem acabou sendo descartado devido a baixa performance nos experimentos. Algumas das falhas encontradas são o grande tempo de latência para a entrega de pacotes de radio quando o sinal se torna marginal, o que o torna inviável para o controle em tempo real do veiculo. Possivelmente uma implementação de rede de sensores, incluindo sensores moveis em VANTs, eliminaram as limitações. Estudos sobre este problema podem ser encontrados em (PIGNATON) e (PEREIRA).

O sistema baseado no *transceiver* Si1000 da empresa HopeRF apresentou resultados melhores que que os obtidos com o Xbee. Um dos fatores que contribuem para a performance deste radio é o fato de utilizar um firmware desenvolvido especificamente para transmissão de dados no protocolo MAVLink , o projeto se encontra em (SiK). Para a conexão deste radio com o dispositivo Android foi utilizado a porta USB disponível no modulo projetado pela empresa 3DRobotics. Um problema foi encontrado com a utilização deste radio, que é a necessidade de ter a antena conectada fisicamente no *tablet* (reduzindo a portabilidade do projeto).

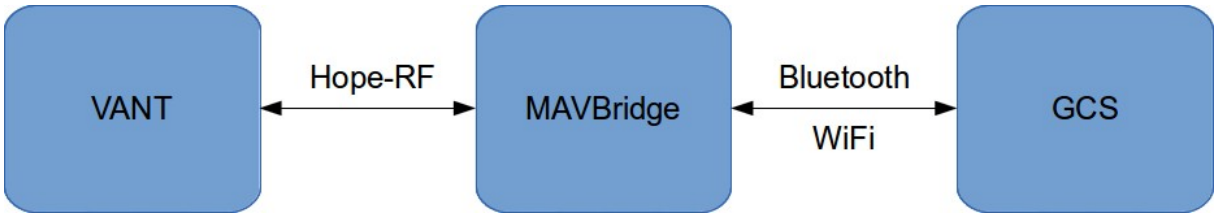
As soluções baseadas em WiFi e Bluetooth foram descartadas inicialmente por não terem o alcance necessário para está aplicação. Porem elas são uteis no caso de se utilizar a estação de controle apenas para a configuração de voos autônomos no VANT.

Nenhuma dos rádios analisados anteriormente acima atingiu todas as metas definidas. Por isso foi projetado um hardware para utilizar os pontos fortes de cada tipo de radio, esse hardware foi chamado de MAVBridge.

1. MAVBridge

Devido aos problemas explicitados anteriormente foi desenvolvida uma placa para utilizar dois tipos de radio. Os rádios utilizados são o modulo 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 fisicamente. Como alternativa ao Bluetooth é possível utilizar WiFi nesta placa. A figura 10 ilustra o funcionamento desta placa mostrando os três componentes envolvidos no link.

Figura 10. Diagrama do link de comunicação da placa MAVBridge

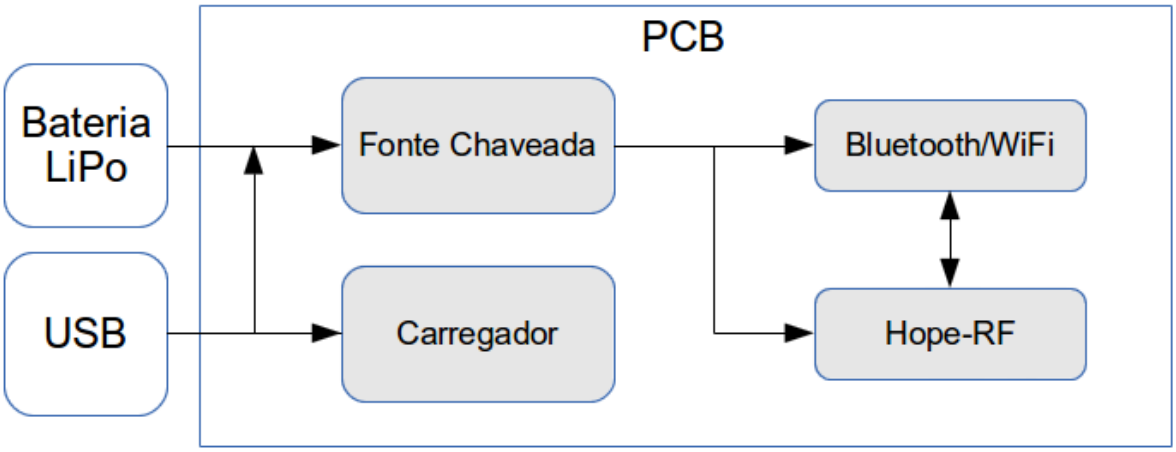


Algumas das vantagens quanto a utilização desta placa são:

- 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 a carga entre as baterias do dispositivo Android e da MAVBridge

A placa pode ser dividida em quatro partes conforme a figura XX. Existe uma conexão direta entre ambos os módulos de radio. E um sistema de alimentação e recarga da bateria.

Figura 11. MAVBridge – Diagrama de Blocos

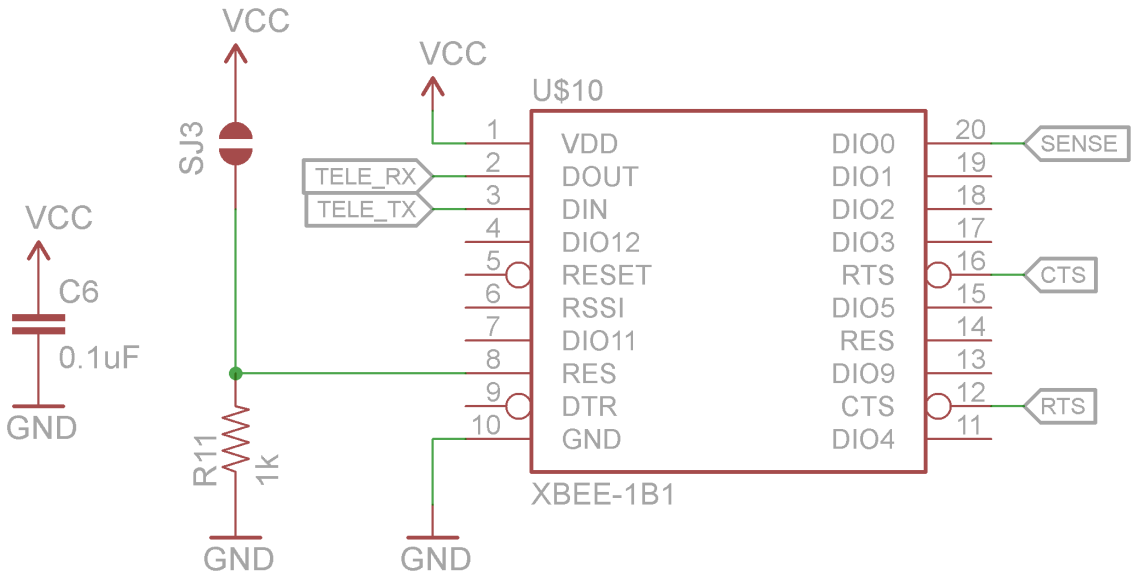


1 Esquemático

O projeto da MAVBridge é relativamente simples pois o design dos circuitos de RF já estão prontos devido ao uso dos módulos da Microchip. Porem o layout deve ser realizado de forma correta para evitar ruídos na fonte de alimentação. O esquemático completo está anexado ao trabalho.

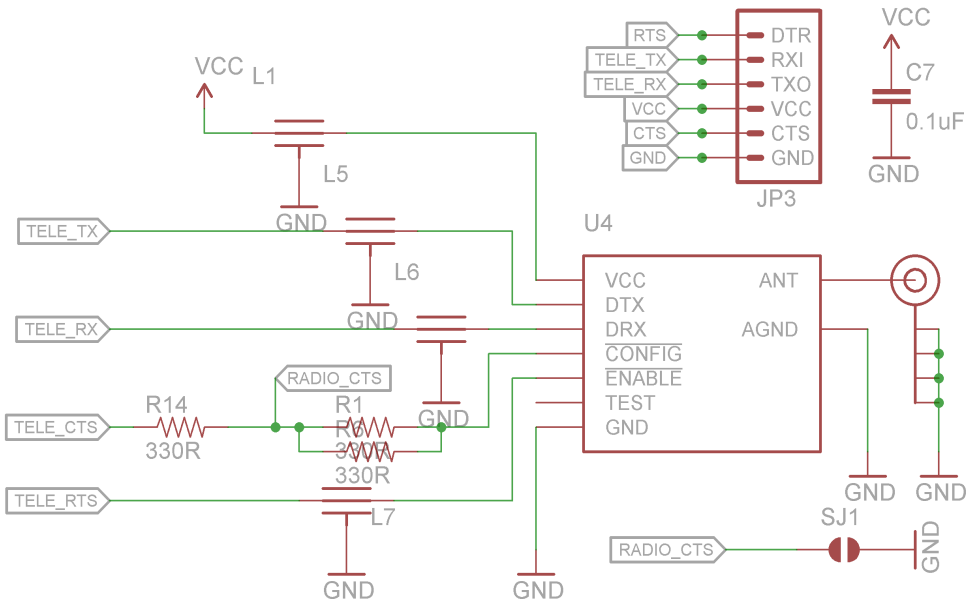
A figura XX mostra a parte do esquemático responsável pelos módulos Bluetooth ou Wifi. Como os módulos RN-42 e RN-171 da empresa Microchip (Figuras XX e XX) tem o mesma pinagem e funcionalidade é possível apenas trocar o modulo que está instalado na placa, isso disponibiliza o uso de tanto WiFi como Bluetooth para o link local. O *jumper* SJ3 serve para reiniciar o modulo com os padrões do fabricante.

Figura 12. MAVBridge – Modulo Bluetooth/Wifi



A figura XX contem o modulo de radio 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 modulo, e para filtrar a linha de alimentação. Uma alimentação sem ruido é essencial para o funcionamento correto deste modulo. O *jumper* SJ1 em conjunto com os resistores R14, R1 e R6 podem colocar o modulo em modo de *bootlooader,* possibilitando a mudança do *firmware* utilizado. O conector JP3 pode ser utilizado caso seja necessário utilizar um modulo diferente do Hope-RF para a saida de telemetria.

Figura 13. MAVBridge – Modulo HopeRF

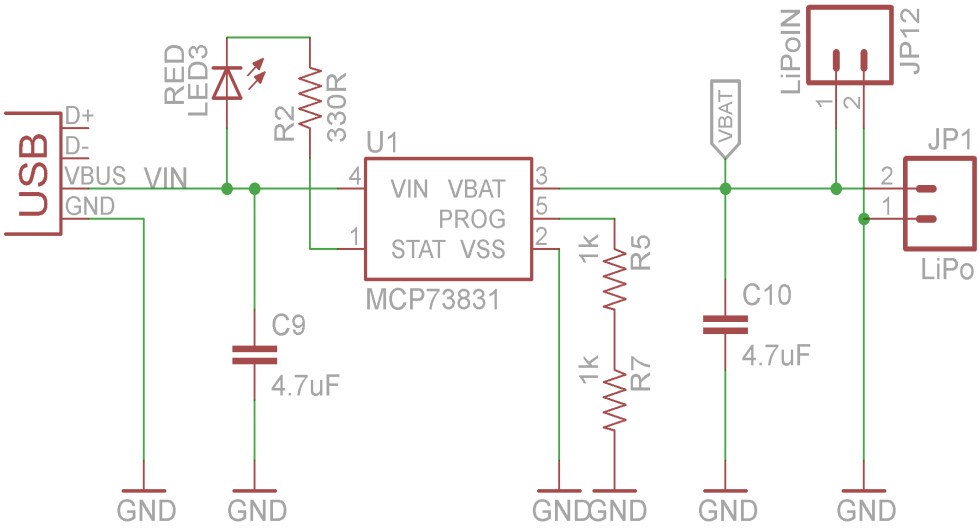


A bateria utilizada no projeto é uma do tipo LiPo, que requer alguns cuidados especiais para ser realizada a carga. Mas como vantagem ela apresenta uma alta densidade energética, e um custo baixo.

O ciclo de carga é realizado pelo circuito da figura XX, o qual é a implementação tipica do C.I. MCP73831 (MICROCHIP). A corrente de carga programada, pelos resistores R7 e R5, é de 500mA o que 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. Pois 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 14. MAVBridge – Carregador de bateria LiPo



A fonte chaveada tem a função de regular a tensão de alimentação do circuito, utilizando a energia armazenada na bateria. Porem a utilização de uma bateria do tipo LiPo (*Lithium Polymer battery*) gera algumas peculiaridades no projeto desta fonte de alimentação.

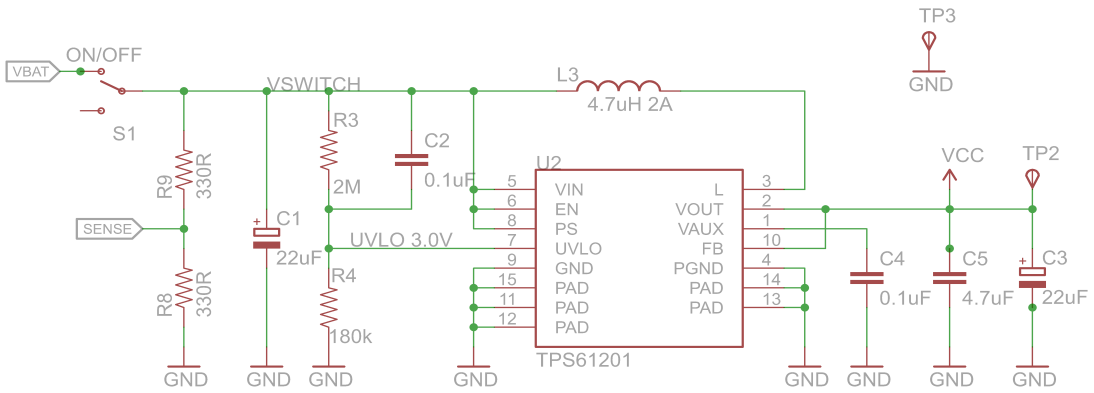
O problema encontrado é que a tensão de alimentação do circuito é fixa em 3,3V, enquanto a tensão da bateria varia entre 4,2 V a 2,75 V (dependendo principalmente da quantidade de carga restante). Logo é necessário a utilização de uma fonte chaveada (para manter a eficiência elevada), porem está tem 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 contrario deve operar como topologia *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 e necessário um circuito de proteção contra sub-tensão na fonte chaveada.

Estas restrições de projeto foram solucionadas pelo uso do C.I. TPS61201 (TEXAS INSTRUMENTS). O qual é indicado especialmente para este 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 XX é o circuito de aplicação tipica 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 servem de ponto de medida de tensão da bateria, o qual pode ser utilizado pelos módulos de WiFi/Bluetooth para

estimar a carga restante. A chave S1 serve de interruptor para o circuito, porem deixa o circuito de carga conectado a bateria, para que seja possível carregar enquanto o circuito está desligado.

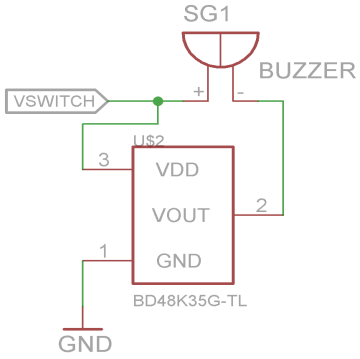
Figura 15. MAVBridge – Fonte Chaveada



O risco de danificar a bateria devido a descarga extrema foi resolvido pelo circuito de UVLO. Porem 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. BD45K35 o qual é utilizado normalmente para proteção de UVLO de circuitos digitais, porem foi reutilizado aqui devido as suas características interessantes. Ele contem uma referencia de tensão estabilizada internamente, e um comparador com saída *open-colector* de alta corrente de saida capaz de acionar diretamente um *buzzer* piezo elétrico. A figura XX mostra o circuito projetado.

Figura 16. MAVBridge – Alarme de bateria baixa



2 Layout

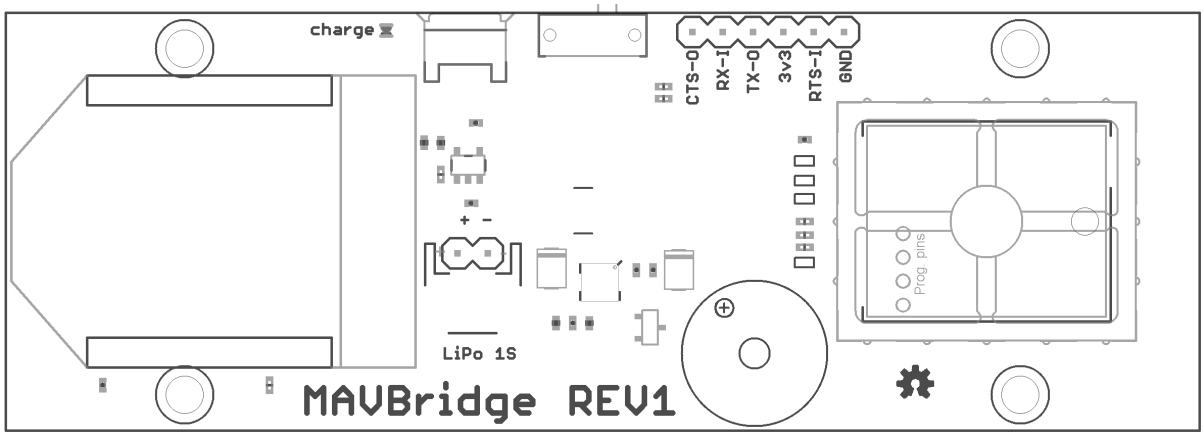
O layout do circuito foi realizado de forma a ser montado em cima da bateria, resultando em um produto compacto. A disposição dos componentes é apresentada na figura XX. As dimensões da placa são de 100x35x5 mm. Quatro furos de 3mm foram colocados nas laterais para fixação.

O modulo 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 modulo montado verticalmente, maximizando a cobertura da antena de telemetria.

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

Na parte superior estão localizados o conector de carga, chave liga/desliga, 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 centra inferior, e a bateria é atras da placa de circuito impresso.

Figura 17. MAVBridge – Layout dos componentes



O layout foi realizado em uma Placa de Circuito Impresso (PCI) de duas camadas, de forma 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 ruido 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 XX apresenta a camada superior de cobre da placa, e a figura XX a camada inferior.

Figura 18. MAVBridge – Layout da camada superior

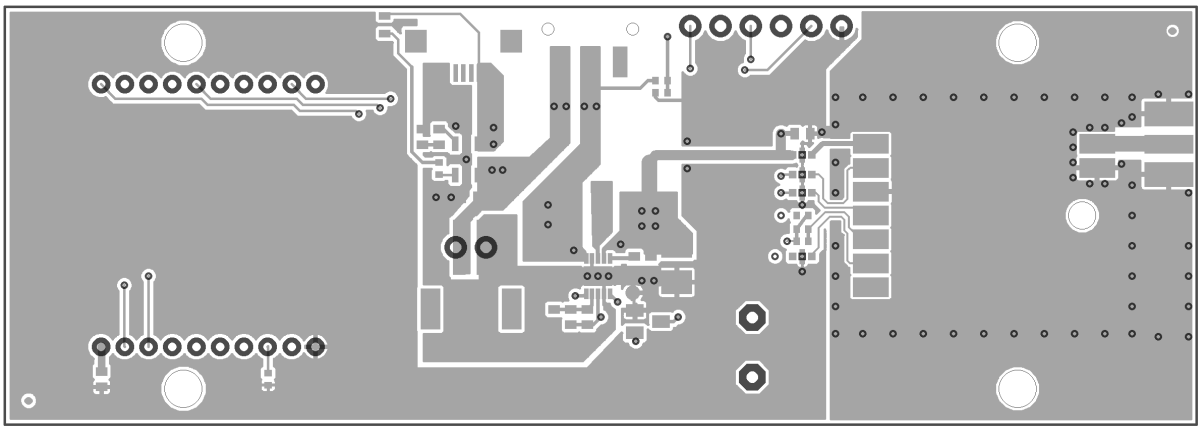
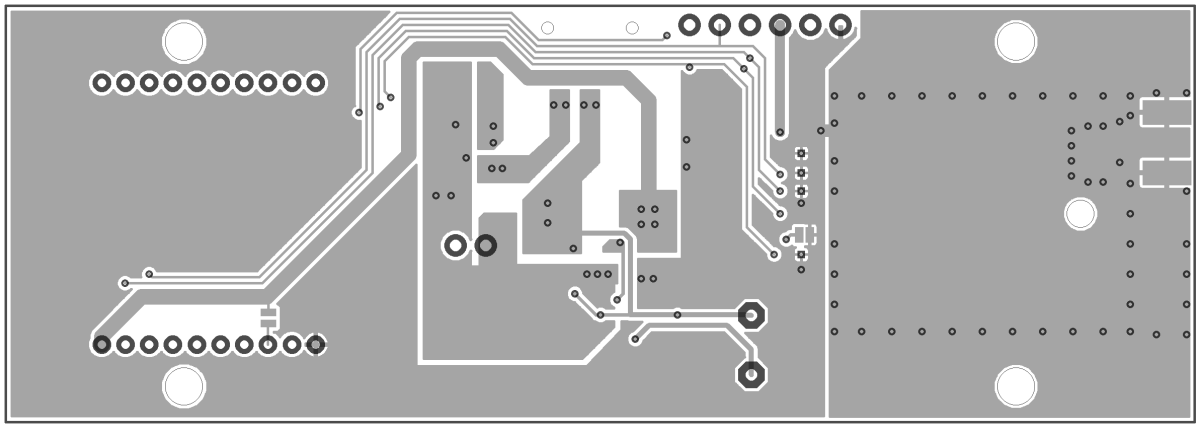


Figura 19. MAVBridge – Layout da camada inferior



2. Conexão direta via USB

Como auxiliar ao MAVBridge o software também oferece suporte a conexão direta a um modulo de radio HopeRF ou Xbee. Porem como a maioria dos dispositivos Android apenas apresenta uma porta USB micro-B um adaptador é necessário, esse é encontrado comercialmente com o nome de cabo *USB-OTG*. A figura XX apresenta um *tablet* rodando o software deste projeto com um modulo de radio contactado.

Figura 20. Radio Hope RF conectado via porta USB através de um adaptador.

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

Figura 21. Modulo de telemetria 3DRobotics

Fonte: 3DRobotics, 2013

5 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.

O sistema operacional escolhido foi o Android, devido as vantagens descritas no capitulo 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 o fato de ser a linguagem principal deste sistema operacional. O *Integrated Development Enviroment* (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 pois essa licença garante que o trabalho de todos no projeto continuara livre.

Para facilitar o desenvolvimento, o qual é realizado pro 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 deste sistema incluem a facilidade de controlar o trabalho de diversos programadores, tornando o processo de integrar o trabalho de todos em uma única fonte "oficial" mais simples. Outra ferramenta muito utilizada é o sistema de rastreamento de *bugs*, o que facilita que usuários reportem problemas na operação do programa.

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

5.1 ARQUITETURA

A arquitetura do software é baseada no modelo *Model-View-Controller* (MVC). MVC é um modelo de arquitetura de software que separa a representação da informação da interação do usuário com ele. As ideias centrais por trás do MVC são a reusabilidade de código e separação de conceitos. (MCV, Wikipedia)

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

A visão (*view*) são 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, e são 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 trafego de telemetria, convertendo-a em comandos para modificar o modelo (que por consequência notifica os componentes de visão). Os dados também trefegam no outro sentido quando uma interface de usuário é acionada, o modelo é modificado e os comandos necessários são enviados ao VANT real. Nesta implementação o controlador está ligado diretamente ao "VANT" virtual, garantindo que as transações de dados (como uma missão autônoma) seja sincronizada 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 abs

Interface de usuário

Meio de comunicação do usuário com a estação de controle

Exibi dados do VANT virtual

Modelo virtual do VANT

Armazena informações localmente do estado do VANT

Transações de missões

Decodifica as mensagens do protocolo MAVLink

Interface de comunicação

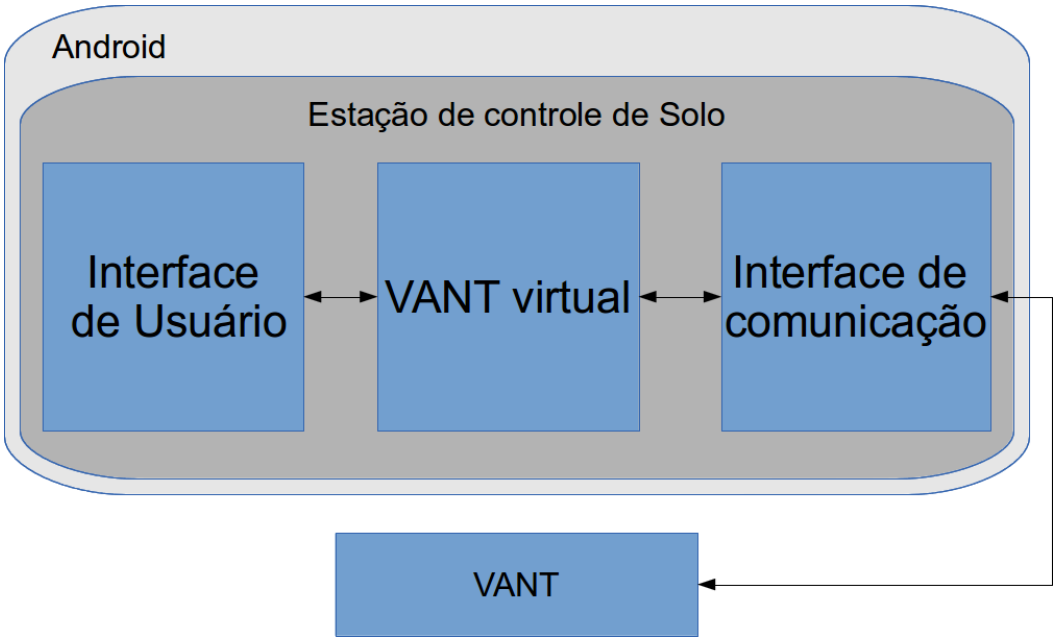
USB – Comunicação direta com Xbee e modulo HopeRF

Bluetooth - MAVBridge

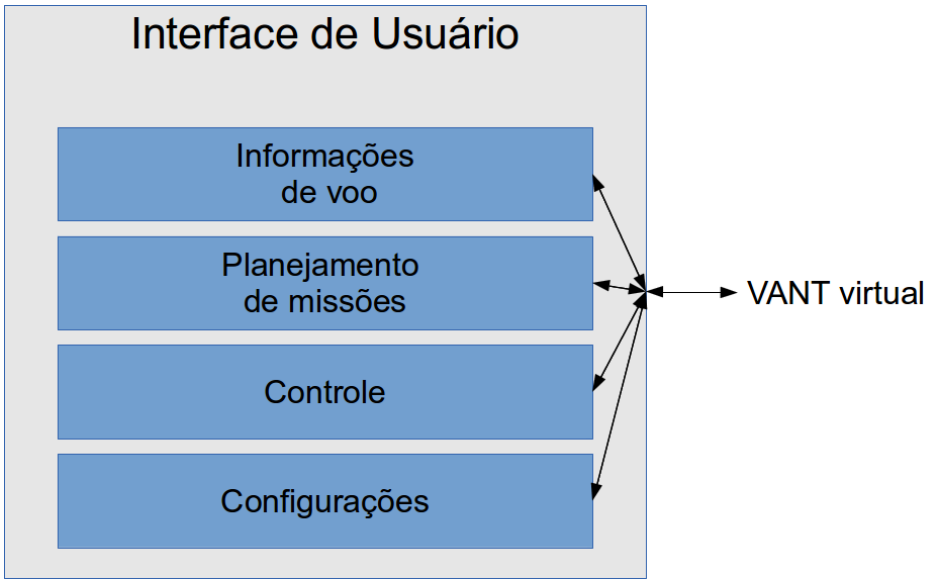
TCP – Link de comunicação 3G

UDP – Link de comunicação WiFi

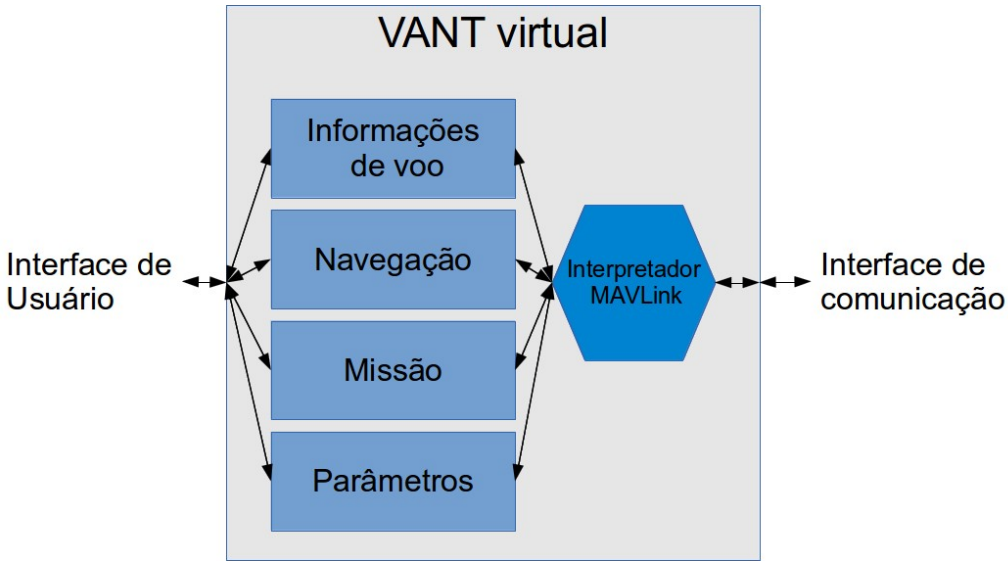
Figura 22. Diagrama de blocos da arquitetura do software



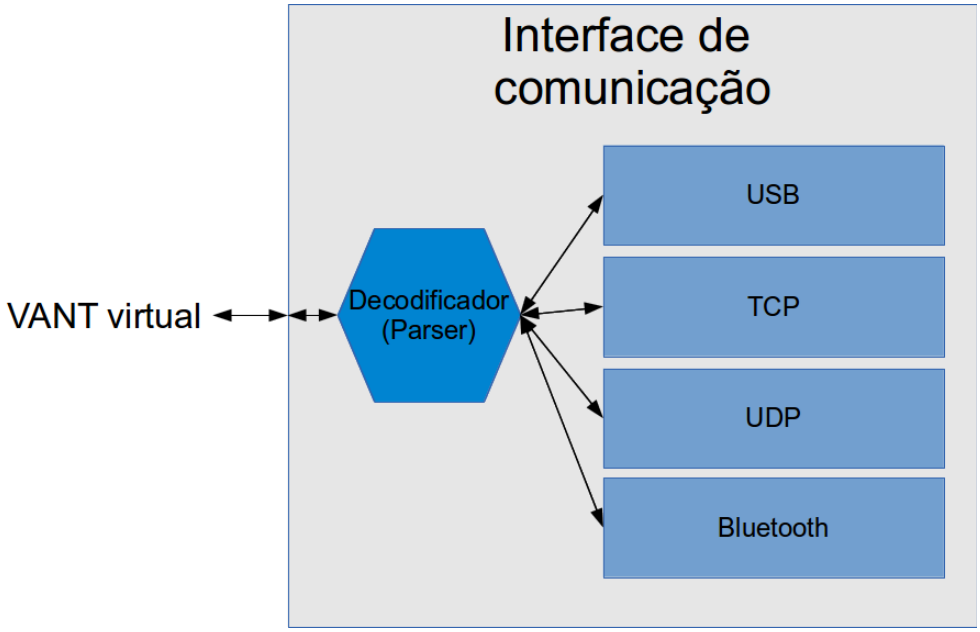
1. **Interface de usuário**



2. **VANT virtual**



3. Meios de Comunicação



USB

Bluetooth

TCP

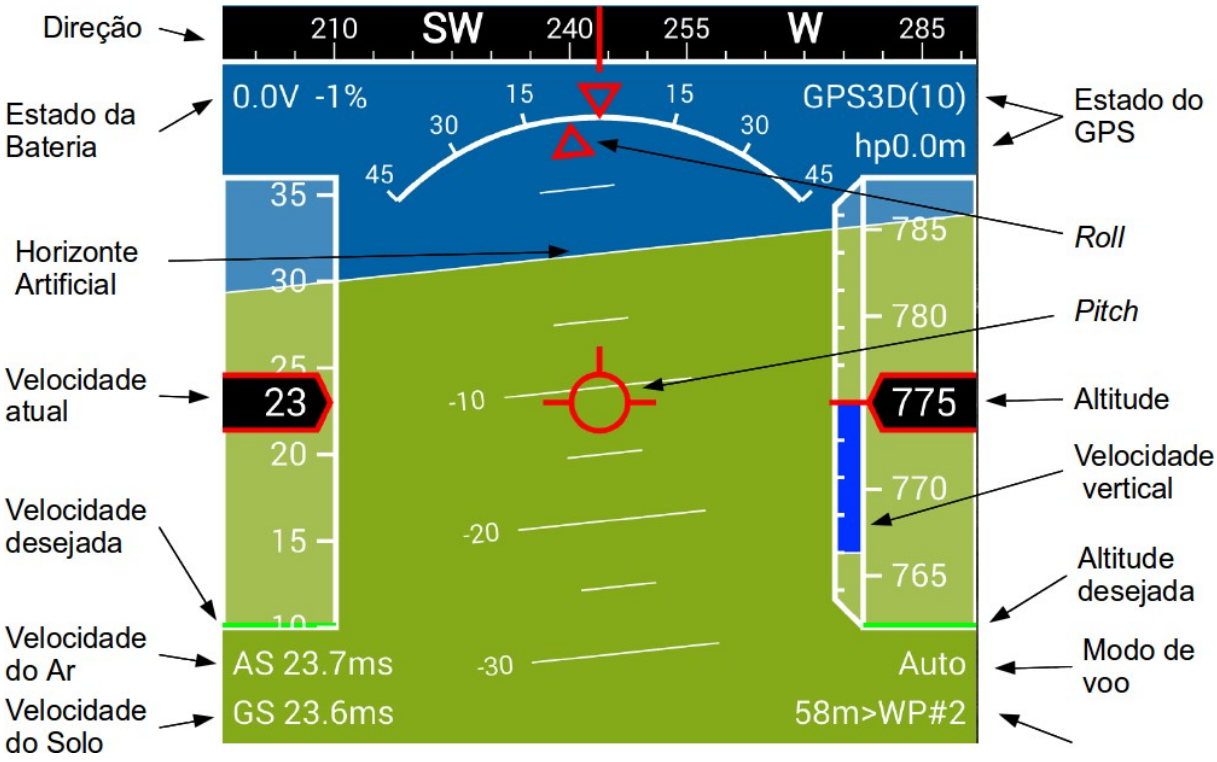
UDP

5.2 INTERFACE HOMEM MAQUINA

1 Informações de voo

Figura 23. Tela de Informações de voo

Figura 24. Detalhes do *Heads Up Display*

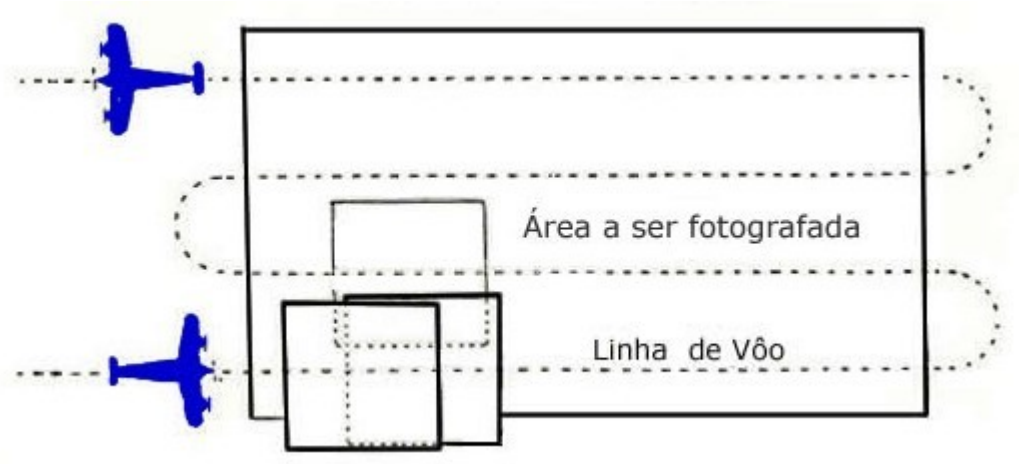


2 Planejamento de missões

Figura 25. Tela de Planejamento de missões

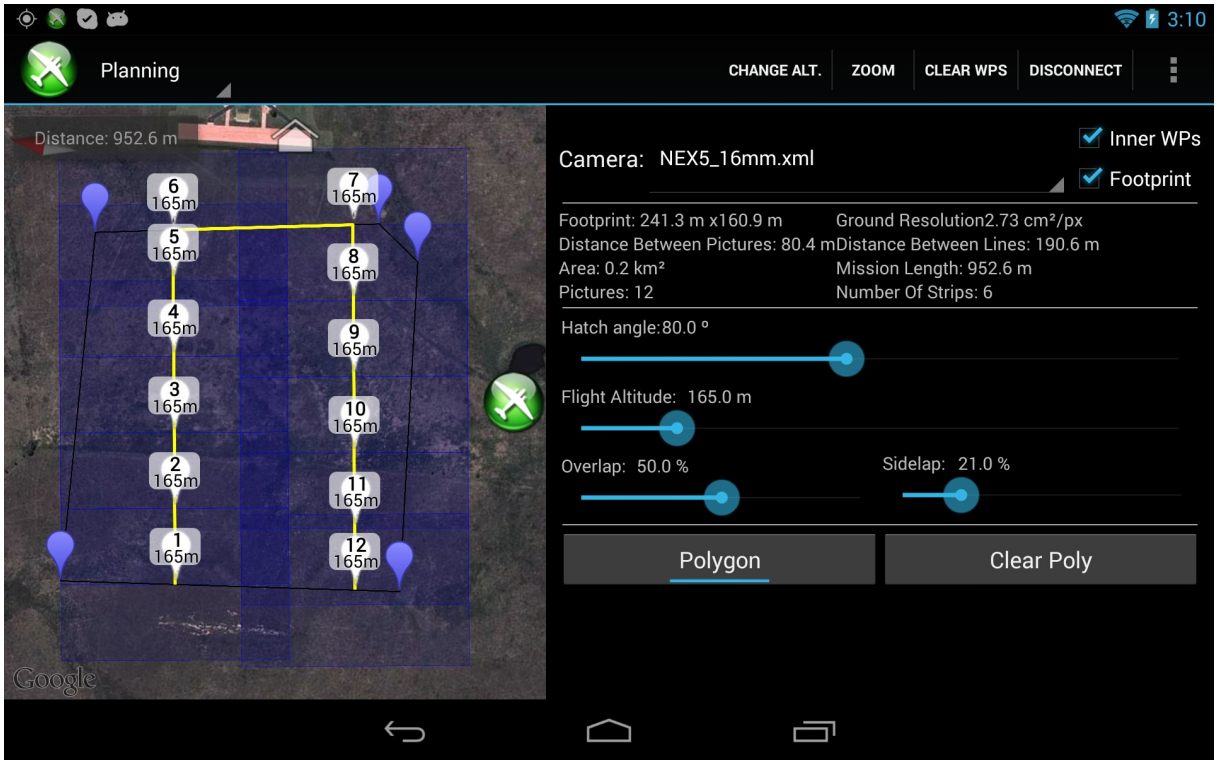
3 Planejamento aerofotogramétrico

Figura 26. Voo fotogramétrico



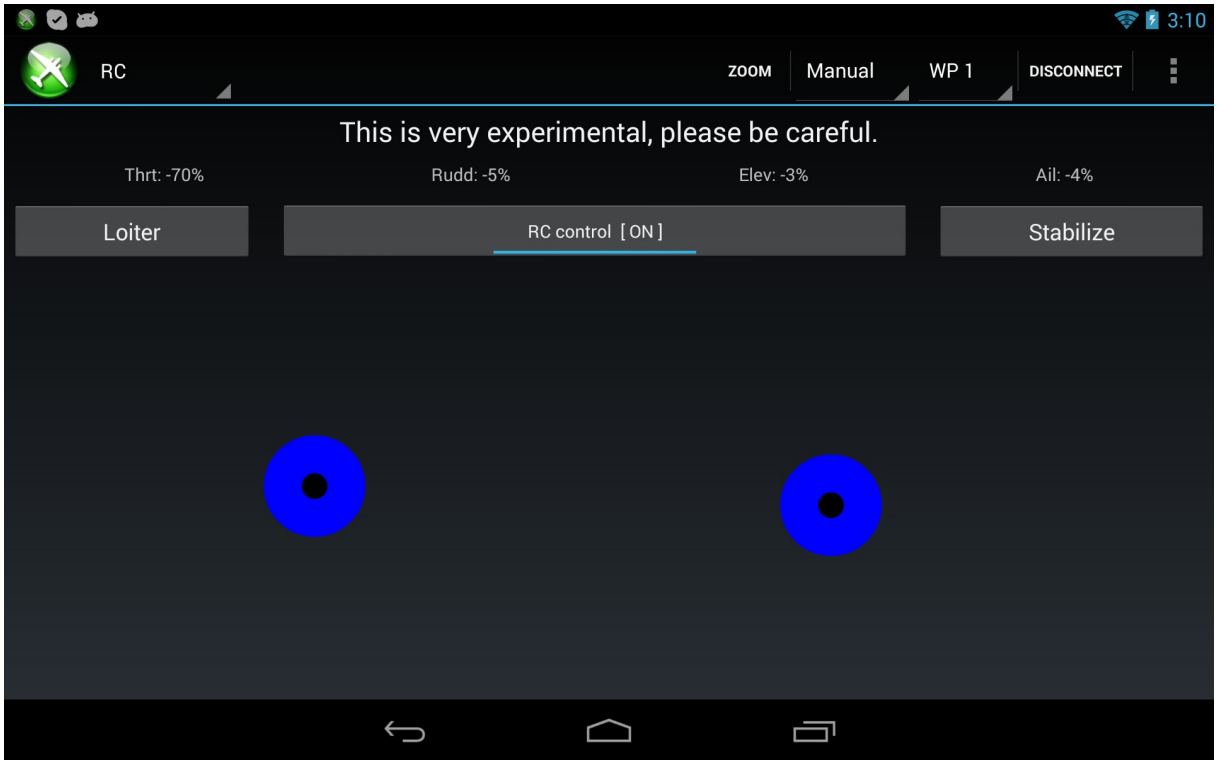
Fonte: UFF, 2013

Figura 27. Tela de Planejamento aerofotogramétrico



4 Controle

Figura 28. Tela de Controle

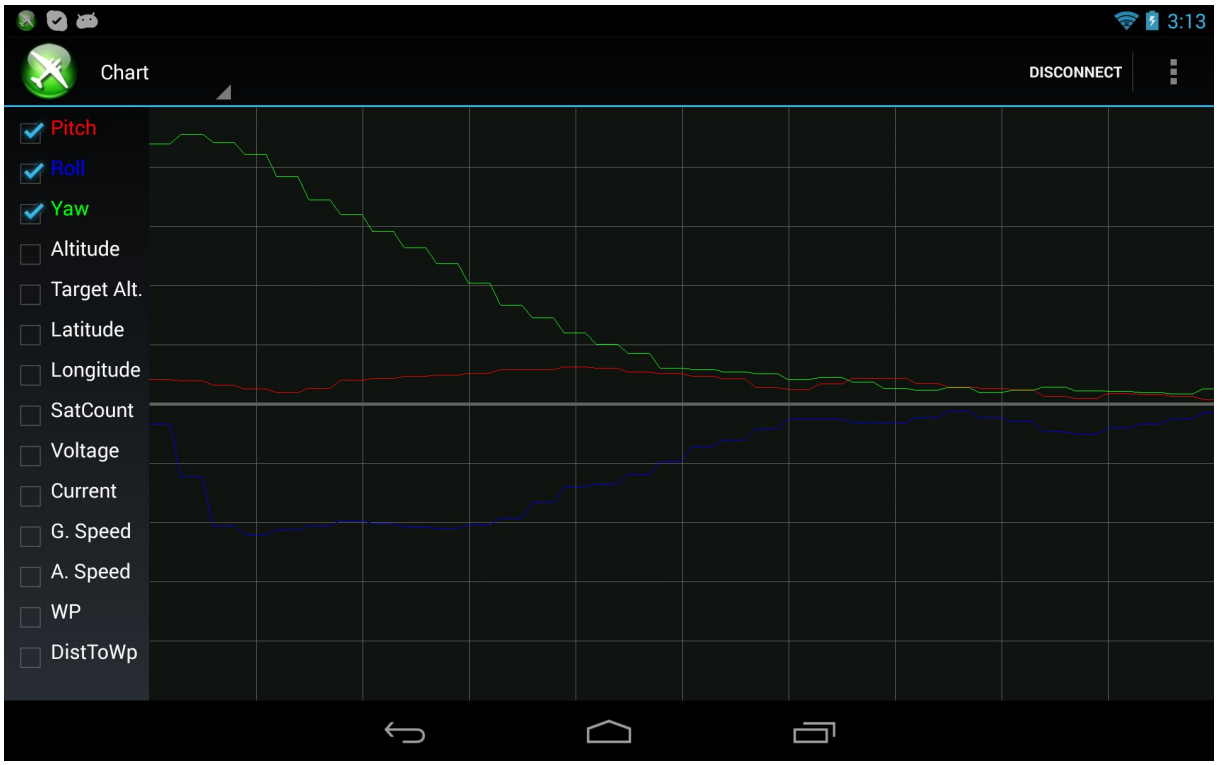


5 Parâmetros

Figura 29. Tela de Parâmetros

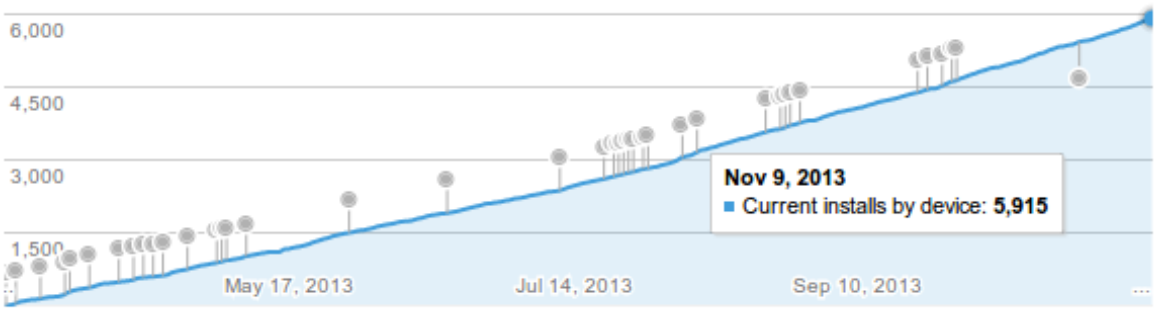
6 Gráficos

Figura 30. Tela de Gráficos


3:13
Chart
DISCONNECT
Pitch
Roll
Yaw
Altitude
Target Alt.
Latitude
Longitude
SatCount
Voltage
Current
G. Speed
A. Speed
WP
DistToWp

6 RESULTADOS

Figura 31. Numero de dispositivos com o aplicativo instal



Dados quantitativos sobre o projeto:

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

7 PROPOSTAS DE MELHORIAS

8 CONCLUSÕES

REFERÊNCIAS

BEARD, R. W.; McLAIN, T. W. **Small Unmanned Aircraft.** First edition. 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.

Athos Alexandre Lima Fontanari, Flavio Rech, and Carlos Eduardo. Sistema de planejamento e controle de missao de um veiculo aereo nao-tripulado. 2011.

PEREIRA, C. E. **Integracao de redes de sensores sem fio com veiculos aereos nao-tripulados (VANTs)**. Premio Santander 2010, 2010.

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

3DRobotics. http://www.3drobotics.com

Skydrones. http://www.skydrones.com.br

DroidPlanner. https://github.com/arthurbenemann/droidplanner

DroidPlanner – Compatible Devices. https://github.com/arthurbenemann/droidplanner/wiki/Compatible-Devices

SiK - https://github.com/tridge/SiK

FREE SOFTWARE FUNDATION - https://gnu.org/licenses/gpl.html

ECLIPSE - http://www.eclipse.org/

ANDROID - http://source.android.com/

ANDROID SDK - http://developer.android.com/sdk/index.html

GIT - http://git-scm.com/

GITHUB - https://github.com/

MCV, Wikipedia - http://en.wikipedia.org/wiki/Model-view-controller