UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CARLOS EDUARDO TUSSI LEITE

Plataforma de controle para Múltiplos Veículos Aéreos Não Tripulados (VANTs)

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2015

RESUMO

Com a perspectiva da regulamentação do uso de VANTs, espera-se um expressivo crescimento do mercado neste setor. O controle de um VANT individualmente é um problema já bem tratado e descrito na literatura, entretanto, o controle de múltiplos VANTs ainda apresenta desafios. A proposta deste trabalho é a extensão de uma plataforma de controle de VANTs chamada DroidPlanner para o controle de múltiplos VANTs simultaneamente. Para realizar a extensão proposta bem como atender às exigências que essa nova aplicação exige, várias mudanças foram feitas em relação ao design original. Estas mudanças podem ser divididas principalmente em três partes. A parte de comunicação, a qual inclui a interação entre a plataforma de controle desenvolvida com os VANTs; A parte das estruturas internas, referente aos principais componentes internos do sistema responsáveis por manter o estado do aplicativo; A parte da interface com o usuário, onde todas as mudanças se refletem. Finalmente, a fim de testar as modificações feitas, foi usado um simulador de VANTs.

Palavras-chave: VANT. controle de múltiplos VANTs. DroidPlanner.

Control Platform for Multiple UAV (Unmanned Aerial Vehicle)

ABSTRACT

With the prospect of regulating the use of UAVs, we expect a significant market growth in this sector. The control of a UAV individually is a problem already well treated and described in the literature, however, the control of multiples UAVs still presents challenges. The purpose of this work is the extension of a UAV control platform named DroidPlanner to control multiple UAVs simultaneously. In order to carry out the proposed extension and also meet the demands that the new application requires, several changes were made to the original design. These changes can be mainly divided into three parts. Part of communication, which includes the interaction between the control platform developed with the UAVs; Part of the internal structures, which refers to the main internal system components responsible for maintaining the state of the application; The part of the user interface, in which all changes are reflected. Finally, in order to test the changes made, a UAV simulator was used.

Keywords: UAV, control multiple UAV, Droid Planner.

LISTA DE FIGURAS

Figura 1.1 Classificação de VANTs em grupos 10

Figura 2.1 Mecanismo de criação de missões - DroidPlanner 2 14
Figura 2.2 Formato de um pacote MAVLink 15
Figura 2.3 Arquitetura do SITL 18

Figura 3.1 Samsung Galaxy Note 10.1 2014 Edition 19
Figura 3.2 3DR IRIS+ 21
Figura 3.3 Módulo XBee - ZB Series 2 utilizado. 21
Figura 3.4 Arquitetura de camada do protocolo ZigBee 23
Figura 3.5 Topologia de uma rede ZigBee 25
Figura 3.6 Comunicação *point-to-multiple* de módulos *XBee* 27
Figura 3.7 Conexão dos módulos ao tablet e ao computador 28

Figura 4.1 Diagrama de sequência de um cenário onde usuário envia um comando. 30
Figura 4.2 Diagrama de sequência representando cenário com mensagens enviadas pelo VANT 31
Figura 4.3 Consulta à tabela hash ao recebimento de uma mensagem 33
Figura 4.4 Diagrama representando a arquitetura MVC modificada 35
Figura 4.5 Diagrama representando o módulo de controle 35
Figura 4.6 Visão interna do modelo de representação de um VANT 36
Figura 4.7 Arquitetura da interface com o usuário para um VANT 37
Figura 4.8 Configuração do Novo Menu de Atividades 38
Figura 4.9 Múltiplas telas de visualização 39
Figura 4.10 Controle e visualização simultânea de 2 VANTs 40
Figura 4.11 Novo conjunto de botões adicionado 41
Figura 4.12 Tela expandida com foco em um único VANT 41
Figura 4.13 Configurando Waypoints 42
Figura 4.14 Criação de Missões 42
Figura 4.15 Menu dos Algoritmos de Reconhecimento 43
Figura 4.16 Visualização das missões criadas 44

Figura 5.1 Interface do simulador utilizado 46
Figura 5.2 Tela inicial da aplicação 47
Figura 5.3 Controle simultâneo de dois VANTs 47
Figura 5.4 Controle simultâneo de três VANTs 48
Figura 5.5 Controle simultâneo de quatro VANTs 48

LISTA DE TABELAS

Tabela 2.1 Definição de mensagens MAVLink 16

Tabela 3.1 Especificação Técnica da Plataforma Android 19
Tabela 3.2 Principais especificações do IRIS+ 20
Tabela 3.3 Especificações Técnicas do Módulo XBee - ZB Series 2 22
Tabela 3.4 Abordagens adotadas para roteamento 26

LISTA DE QUADROS

2.1 Exemplo de definição de uma mensagem MAVLink. 16

4.1 Definição da nova mensagem MAVLink criada. 33

LISTA DE ABREVIATURAS E SIGLAS

APM	Ardu Pilot Mega
GCS	Ground Control Station
LGPL	Lesser General Public License
MAV	Micro Air Vehicle
MVC	Model-View-Controller
SMT	Surface Mounted Components
SITL	Software in the Loop
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VANT	Veículo Aéreo Não Tripulado
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Contextualização 10
1.2 Objetivo 11
1.3 Organização do Trabalho 11
2 REVISÃO DE CONCEITOS E TRABALHOS RELACIONADOS 13
2.1 Droidplanner 13
2.2 Protocolo MAVLink 14
2.2.1 Pacotes MAVLink 15
2.2.2 Definição de mensagens MAVLink 15
2.3 Simulador 17
3 FERRAMENTAS 19
3.1 Plataforma de Controle 19
3.2 VANT 20
3.3 Comunicação 20
3.3.1 XBee 22
3.3.2 ZigBee 22
3.3.2.1 Coordenador 23
3.3.2.2 Roteadores 24
3.3.2.3 Dispositivos finais 24
3.3.2.4 Endereçamento e transmissão 25
3.3.3 Configuração dos Módulos XBee 26
3.3.3.1 Geração da Rede 27
4 IMPLEMENTAÇÃO 29
4.1 Etapas 29
4.1.1 Familiarização com os componentes preexistentes 29
4.1.2 Link de Comunicação 31
4.1.3 Estruturas Internas 32
4.1.4 Definição de nova mensagem MAVLink 33
4.1.5 Modificações na Interface 34
4.2 Arquitetura 34
4.2.1 Controle 34
4.2.2 Modelo 35
4.2.3 Visão 36
4.3 Interface com Usuário 37
4.3.1 Comando e Visualização 38
4.3.1.1 Múltiplas telas de Visualização 39
4.3.1.2 Controle e visualização de informação dos VANTs 40
4.3.2 Planejamento de Missões 41
4.3.3 Visualização de Missões 43
5 EXPERIMENTOS 45
6 CONCLUSÃO 49
6.1 Trabalhos futuros 49
6.1.1 Prevenção de colisões 49
6.1.2 Comunicação entre VANTs 50
6.1.3 Múltiplos canais de Comunicação 50
REFERÊNCIAS 51

1 INTRODUÇÃO

Veículos aéreos não tripulados, doravante denominados simplesmente VANTs, estão cada vez mais se tornando populares não apenas para uso militar, para o qual foram originalmente idealizados, mas atualmente também para uso civil. Seja para uso em resgates em locais de difícil acesso, captura de fotos e vídeos ou para quaisquer outros fins alternativos, o uso dessas aeronaves está se tornando cada vez mais relevante.

Além disso, com a perspectiva de regulamentação do uso desses veículos, é prevista a expansão do mercado de aplicações civis nos próximos anos, demandando, consequentemente, sistemas digitais capazes de atender essa demanda.

1.1 Contextualização

Veículos aéreos não tripulados surgiram por volta do século XX como resultado de pesquisas militares, para atuarem como veículos de guerra. Com o passar do tempo, ao passo que novas tecnologias foram sendo desenvolvidas, VANTs de diferentes tipos e tamanhos foram projetados para diferentes fins.

Apesar de não haver uma única classificação (ABDULLAH, Q. A, 2014), dada a grande variedade de VANTs existentes hoje em dia, estes podem ser categorizados de acordo com diferentes critérios. Adotando a classificação do Departamento de Defesa dos Estados Unidos (DOD, 2011), VANTs podem ser divididos em cinco grupos de acordo com tamanho, peso, altitude e velocidade. A Figura 1.1 ilustra essa classificação.

O grupo de pequenos VANTs, na sua grande maioria, é responsável pela maior parte das aplicações no âmbito civil. Estas, por sua vez, englobam os mais variados

Figura 1.1 – Classificação de VANTs em grupos

Category	Size	Maximum Gross Takeoff Weight (MGTW) (lbs)	Normal Operating Altitude (ft)	Airspeed (knots)
Group 1	Small	0-20	<1,200 AGL*	<100
Group 2	Medium	21-55	<3,500	<250
Group 3	Large	<1320	<18,000 MSL**	<250
Group 4	Larger	>1320	<18,000 MSL	Any airspeed
Group 5	Largest	>1320	>18,000	Any airspeed

Fonte: (ABDULLAH, Q. A, 2014)

contextos. Sua utilização em áreas como a agricultura de precisão (AMATO, A, 2014), fotografia (PHANTOM, 2015), mineração (SHELLBORN, A, 2015), serviços de emergência (MICRODRONES), e, atualmente, utilização no segmento de entrega de produtos como anunciado pela empresa Amazon (AMAZON, 2014) e pela empresa Domino's para entrega de pizza (PEPITONE, J. , 2013), impulsionam o mercado de VANTs de pequeno porte. Entretanto, na proporção em que a variedade de aplicações crescem, novas tecnologias e soluções são necessárias para dar suporte a esses diferentes contextos de uso.

Plataformas para controle de apenas um VANT já possuem diversas soluções conhecidas como, por exemplo, Mission Planner (MISSIONPLANNER), Andro Pilot (ANDROPILOT), DroidPlanner (DROIDPLANNER), entre outras aplicações, já sendo bem discutidas e tratadas. Por outro lado, ao se trabalhar com mais de um veículo deste tipo simultaneamente, são necessárias novas abordagens para solucionar toda uma nova gama de problemas decorrentes do uso de múltiplo VANTs como, por exemplo, prevenção de colisões entre os veículos, comunicação desses veículos com a plataforma de controle, comunicação dos próprios veículos entre si, suporte de uma plataforma de controle adequada para visualização, além de outras questões.

Finalmente, a utilização de diversos VANTs para um objetivo comum possibilita aplicações mas sofisticadas como, por exemplo, redes *ad-hoc* criadas por diversos VANTs (ALSHBATAT, A. L., 2010), entre muitas outras. Com isso, técnicas que deem suporte a esse tipo de aplicação são imprescindíveis.

1.2 Objetivo

Levando em consideração a atual situação e necessidade do mercado de VANTs, este trabalho tem por objetivo desenvolver um controlador para múltiplos veículos aéreos não tripulados a partir da extensão de uma plataforma já desenvolvida e já bem conhecida para controlar apenas um VANT chamada DroidPlanner (DROIDPLANNER).

1.3 Organização do Trabalho

No capítulo 2 será apresentada a plataforma DroidPlanner, contendo uma breve introdução de sobre seu funcionamento e funcionalidades. Ainda neste capítulo, serão apresentados alguns conceitos importantes para o desenvolvimento do projeto.

O capítulo 3 descreverá as ferramentas utilizadas para o trabalho. Este capítulo se divide em três seções: a primeira se refere à plataforma Android utilizada, a segunda referente à comunicação dessa plataforma com os VANTs e a terceira aos VANTs utilizados para experimentação.

No capítulo 4 serão abordados aspectos relacionados à implementação. Além disso, as principais modificações tanto na interface quanto na arquitetura serão abordadas nessa seção.

Capítulo 5 relatará os experimentos feitos com o resultado da plataforma estendida, bem como algumas questões relevantes sobre o desempenho geral do sistema. Finalmente, capítulo 6 conterá um fechamento para o projeto, bem como propostas de trabalhos futuros.

2 REVISÃO DE CONCEITOS E TRABALHOS RELACIONADOS

2.1 Droidplanner

DroidPlanner é uma estação de controle, ou também chamada de *Ground Controle Station* (GCS), para controlar VANTs, desenvolvida por (BENEMANN, 2013). A primeira versão desse software surgiu em 2013 (DPV1, 2013), estando atualmente na sua terceira versão chamada Tower (TOWER, 2015), esta oferecida pela empresa 3D Robotics. Contabilizando as três versões, o aplicativo DroidPlanner já foi baixado mais de 70 mil vezes, segundo a *Google Play*, loja online para *download* de aplicativos.

O DroidPlanner foi desenvolvido utilizando a licença *GNU General Public License*, o que permite ao usuário modificar e compartilhar o software. Por essa razão, outros desenvolvedores foram adicionados ao projeto ao longo do tempo, auxiliando seu desenvolvimento. Atualmente, todos os códigos estão disponíveis, podendo ser encontrados através da plataforma *Git Hub* (DPGIT), sendo acessíveis a qualquer programador que tenha interesse.

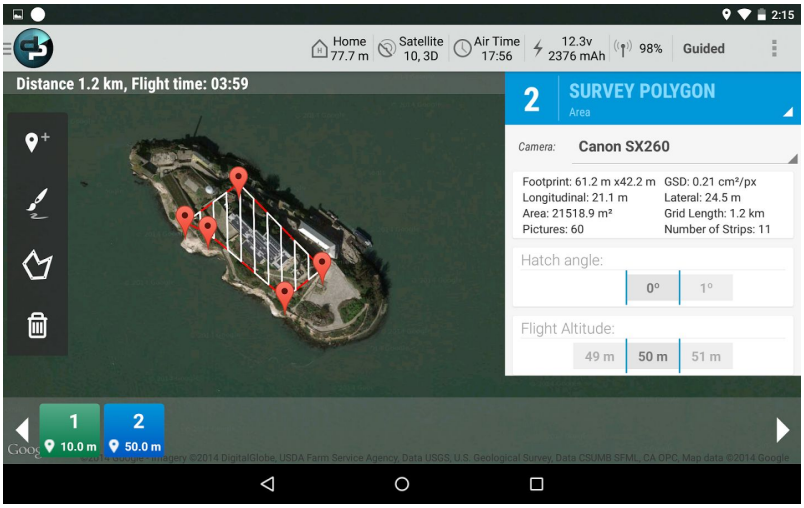
Um dos principais motivos para a escolha do aplicativo DroidPlanner para este projeto se deve justamente pelo fato do código ser aberto e a licença permitir alterações e compartilhamento do mesmo. Além disso, sendo DroidPlanner uma estação de controle já bem conhecida e muito utilizada, já há uma comunidade de programadores que discutem soluções e dúvidas a respeito dessa plataforma.

A versão utilizada para expansão foi a versão 2.8.6 RC3 (DPV286), sendo esta o último *release* da versão 2. O motivo pelo qual foi escolhida a versão 2 ao invés da versão 3 se deve ao fato de que esta roda sobre serviços oferecidos pela empresa 3D Robotics, fazendo com que alguns módulos antes presentes no código DroidPlanner e necessários para a expansão do projeto, encontrem-se nos pacotes oferecidos pela empresa. Além disso, em relação à interface e à funcionalidade, não houve mudanças drásticas, o que contribuiu para a escolha da versão 2.8.6 RC3. A partir do estado final dessa versão, como ponto de partida, foram realizadas as mudanças necessárias para o controle de múltiplos VANTs simultaneamente. Em relação às estruturas internas, estas foram todas estendidas para armazenar as informações de diversos veículos. Já na interface, os elementos já existentes foram todos adaptados para suportar múltiplas visualizações.

O funcionamento básico do aplicativo consiste na troca de mensagens entre um dispositivo móvel e um VANT, utilizando o protocolo MAVLink. As mensagens enviadas

do DroidPlanner para o VANT são ações que o usuário comanda ao veículo. As mensagens enviadas pelo VANT compõem um conjunto de informações que indicam seu estado atual. Além de comandos, a plataforma de controle tem a possibilidade de criar missões, definindo pontos de interesse no mapa a serem visitados pelo veículo. O VANT, por sua vez, recebe estes comandos e os executa, enviando *feedback* de suas ações para o dispositivo móvel a fim de que este possa atualizar seu estado refletido para o usuário através da interface gráfica. A Figura 2.1 ilustra a interface para a criação de missões.

Figura 2.1 – Mecanismo de criação de missões - DroidPlanner 2



Fonte: (GOOGLESTORE, 2014)

A comunicação entre o DroidPlanner e o veículo pode ser realizada de diversas maneiras. Há possibilidade de se usar protocolo UDP através do conexão WiFi, TCP utilizando tecnologia 3G, Bluetooth e também USB, com um módulo de rádio acoplado, por exemplo.

Em relação aos dispositivos alvo, DroidPlanner foi desenvolvido para executar em plataformas rodando o sistema operacional Android 4.0 ou superior. Uma lista de dispositivos compatíveis pode ser encontrada no repositório Git do desenvolvedor (DEVICES).

2.2 Protocolo MAVLink

O protocolo de comunicação utilizado neste trabalho foi o protocolo MAVLink (*Micro Air Vehicle Link*) (MAVLINK). Este protocolo foi desenvolvido em 2009 por Lorenz Meler, utilizando a licença LGPL. A versão implementada no projeto foi a versão 1.0, correspondendo, por motivos de compatibilidade, à versão utilizada pelo DroidPlan-

ner original. Além disso, esta versão já é implementada em outros sistemas que interagem com VANTs como, por exemplo, módulos de piloto automático PIXHAWK (PIXHAWK) e ArduPilotMega (APM).

MAVLink é um protocolo de comunicação muito utilizado por pequenos veículos aéreos não tripulados. Por ser um protocolo leve, é ideal para trocar pequenas informações entre a plataforma de estação de controle desenvolvida e o VANT, não sendo necessário um processamento mais custoso para a manipulação dessas mensagens.

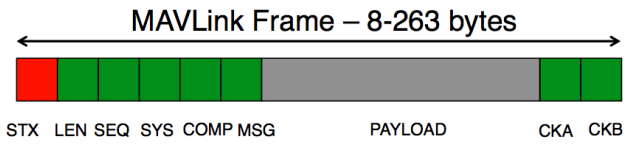
2.2.1 Pacotes MAVLink

Um pacote MAVLink possui estrutura semelhante à Figura 2.2 em que um *frame* é representado por um conjunto de campos obrigatórios de tamanho fixo (1 byte) e um campo de *payload* opcional contendo o conteúdo da mensagem sendo transmitida. A Tabela 2.1 contém os detalhes de cada elemento do pacote bem como o tamanho de cada campo.

Os pacotes podem ter tamanhos diferentes, dependendo do tamanho da mensagem carregada. O conteúdo da mensagem depende do tipo de mensagem sendo transmitida, esta especificada no campo de identificação da mensagem.

Os dois últimos campos do pacote correspondem aos bytes responsável por realizar *CRC*. Essa técnica permite o receptor detectar erros que possam eventualmente ter ocorridos durante a transmissão e descartar pacote corrompido, visto não ser possível recupera-lo.

Figura 2.2 – Formato de um pacote MAVLink



Fonte: (MAVLINK)

2.2.2 Definição de mensagens MAVLink

Cada mensagem MAVLink é identificada através de seu campo identificador presente no cabeçalho do pacote e tem seu conteúdo embutido no campo de dados *payload*,

Tabela 2.1 – Definição de mensagens MAVLink

Byte Index	*Conteúdo*	*Valor*	*Decrição*
0	Sinal do Início do Pacote	0xFE	Indica o início do novo pacote
1	Tamanho do payload	0-255	Indica o tamanho que o campo payload ocupa
2	Número de sequência do pacote	0-255	Permite a detecção da perda de pacotes. Cada componente contabiliza seu próprio número de sequência
3	ID do Sistemas	1-255	ID do sistema emissor da mensagem. É o que permite diferenciar emissores em uma mesma rede.
4	ID do componente do sistema	0-255	ID do componente emissor. Permite diferenciar componentes diferentes no mesmo sistema.
5	ID da mensagem	0-255	ID que identifica que tipo de mensagem está sendo transmitida.
6 até (n+6)	Payload	0-255 bytes	Dados da mensagem.
(n+7) até (n+8)	Checksum	1-2 bytes	Detecção de mensagens corrompidas.

como dito anteriormente. Além disso, essas mensagens estão definidas logicamente através de um documento XML, o qual permite representar os tipos de dados sendo transmitidos, bem como a ordem em que a aplicação deve interpretar os bytes recebidos.

Ao receber uma mensagem semelhante à listada no Quadro 2.1, a camada de aplicação inicialmente verifica o id da mensagem para, após, saber como processar os campos contidos na área de *payload*.

Quadro 2.1 – Exemplo de definição de uma mensagem MAVLink.

```
<message id="49" name="GPS_LOCAL_ORIGIN_SET">
                <description>Once the MAV sets a new GPS−Local correspondence, this
                    message announces the origin (0,0,0) position</description>
                <field type="int32_t" name="latitude">Latitude (WGS84), expressed as * 1E7
                    </field>
                <field type="int32_t" name="longitude">Longitude (WGS84), expressed as * 1
                    E7</field>
                <field type="int32_t" name="altitude">Altitude(WGS84), expressed as * 1000
                    </field>
            </message>
```

O protocolo MAVLink também permite especificar novas mensagens caso seja necessário para a comunicação entre aplicações. Para definir uma nova mensagem, basta representá-la em um arquivo XML, respeitando o padrão estabelecido semelhante ao do

trecho de código acima. Caso seja necessário para aplicação, este arquivo XML pode ser usado para gerar código de forma automática, utilizando alguma das ferramentas disponíveis para esse fim. Para este trabalho, foram utilizadas rotinas do próprio DroidPlanner para gerar os pacotes.

Em relação ao número de mensagens possíveis de serem criadas, apesar do tamanho do campo para identificar os tipos de mensagem ocupar 1 byte, apenas o intervalo compreendido entre 150 e 240 é indicado para novas definições, estando os demais valores reservados para mensagens já pré-definidas.

2.3 Simulador

A fim de validar as modificações feitas na implementação ao longo do desenvolvimento do projeto, foi necessário utilizar um simulador para testes, visto ser importante um ambiente seguro para as experimentações. Além disso, seria importante utilizar um simulador cujo protocolo de comunicação fosse semelhante ao utilizado pelos VANTs físicos. A fim de atender a esses pré-requisitos, foi utilizado o modelo de simulação *Software in the Loop* (SITL).

A arquitetura do sistema possui estrutura semelhante à Figura 2.3. O componente de cor vermelha representa o *software ArduPilot*, responsável pela simulação do comportamento do veículo. Os módulos em azul correspondem às rotinas responsáveis pela dinâmica do voo. As saídas geradas a partir desses dois componentes são mensagens MAVLink, recebidas pela estação de controle sendo utilizada, na imagem representada pelos componentes restantes (opcionais e que na imagem correspondem a diferentes GCS).

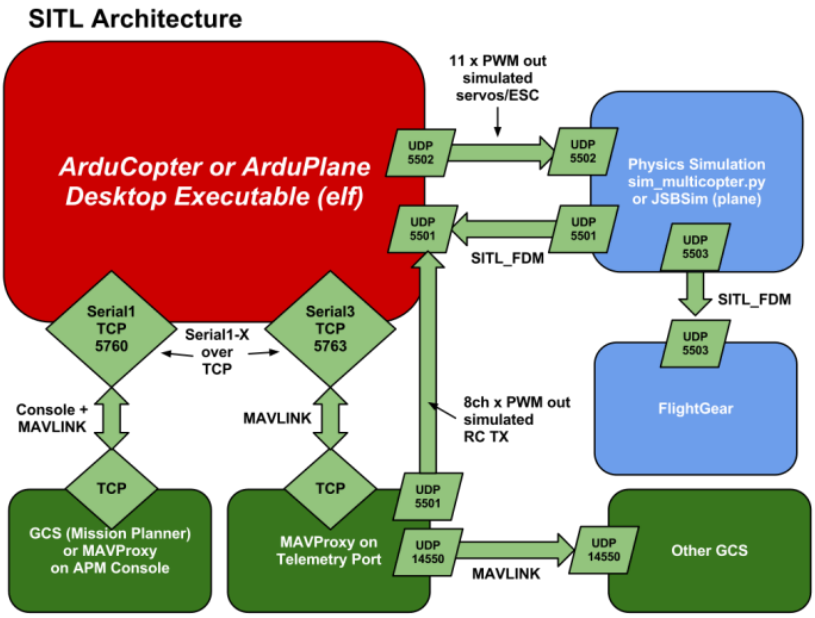
O simulador permite que diferentes tipos de VANTs possam ter seu comportamento simulado de forma precisa, sem a necessidade do *hardware*. Isso ocorre pois SITL utiliza o *software* ArduPilot (ARDUPILOT), programa sofisticado utilizado como piloto automático dos veículos. Em relação ao ambiente de execução, é possível rodar em ambientes *Linux* e *Windows*.

Considerando o contexto deste trabalho, a comunicação entre o aplicativo desenvolvido e a simulação corresponde à ligação entre *Other GCS* e *MAVProxy*. Neste canal, são trocadas as mensagens utilizando o protocolo MAVLink fundamental para o projeto como discutido anteriormente.

Finalmente, em relação ao processo de simulação, basicamente consiste na troca de mensagens MAVLink entre a plataforma de controle e o simulador. Com as mensagens

recebidas por este é possível obter o estado do aplicativo e, então, validar as implementações feitas na estação de controle DroidPlanner.

Figura 2.3 – Arquitetura do SITL



Fonte: (SITL)

3 FERRAMENTAS

3.1 Plataforma de Controle

A plataforma de desenvolvimento escolhida para servir como estação de controle foi o tablet Galaxy Note 10.1 2014 Edition da empresa Samsung. O pré-requisito principal para a escolha deste modelo foi ser um dispositivo Android, além de estar presente na lista de modelos que suportam o aplicativo Droidplanner.

O sistema operacional usado na plataforma foi o Android versão 4.3, também chamado *Jelly Bean*. A Tabela 3.1 contém a especificação técnica dos principais componentes do sistema. A Figura 3.1 mostra uma foto do tablet escolhido.

Tabela 3.1 – Especificação Técnica da Plataforma Android

Sistema Operacional	Android 4.3
CPU	Quad core 1.9GHz + Quad core 1.3 GHz
Memória RAM	3GB LPDDR3
Conectividade	WiFi (802.11 a/b/g/n/ac), BT4.0
Bateria	8.220 mAh
Tela	10.1"TFT WQXGA (2560x1600) pixels

Figura 3.1 – Samsung Galaxy Note 10.1 2014 Edition

Fonte: (SAMSUNG)

3.2 VANT

Apesar dos experimentos deste projeto acontecerem apenas em ambientes simulados, o objetivo final é o controle de VANTs reais. Dessa forma, como modelo de veículos a serem usados nas simulações, foram escolhidos os dispositivos multirotores.

Multirotores, ou multicópteros, são veículos aéreos de mecânica simples controlados aumentando ou diminuindo a rotação de seus múltiplos rotores. Por serem dinamicamente instáveis, obrigatoriamente necessitam de um controlador de voo acoplado, combinando informações de sensores como girosópio, acelerômetro, entre outros, a fim de se manterem estáveis durante o voo.

Por ser necessária a realização de voos autônomos, ou seja, sem o controle direto do usuário na performance, sendo classificando, portanto, como um VANT, o equipamento escolhido deveria ser capaz de combinar diversas informações de voos, como GPS, giroscópio, acelerômetro entre outras, e processar através de seu computador de bordo ou piloto automático.

Atendendo a esses requisitos, optou-se pelo dispositivo multirotor IRIS+. A Figura 3.2 representa a forma do veículo. Os principais componentes desse VANT podem ser resumidos na Tabela 3.2.

Tabela 3.2 – Principais especificações do IRIS+

Peso	1282g
Capacidade de carga	400g
Tempo de vôo	1 16-22 min*
GPS	3DR uBlox GPS with Compass (LEA-6H module, 5 Hz update)
Autopilot	Pixhawk v2.4.5
Firmware	ArduCopter 3.2
Bateria	3S 5.1 Ah 8C lithium polymer

*Tempo de vôo depende da quantidade de carga extra, vento, temperatura e demais fatores físicos

3.3 Comunicação

A comunicação entre a estação de controle e o VANT é uma parte importante para a aplicação, visto que é por onde todas as mensagens MAVLink, contendo todas as informações e comandos necessários para o correto funcionamento do *software* são transmitidas. Uma comunicação eficiente, dentro do contexto da aplicação, é, portanto, fundamental para o sucesso do projeto proposto.

Figura 3.2 – 3DR IRIS+

Fonte: (IRIS)

Como a principal característica desse trabalho é o fato de ser possível controlar múltiplos VANTs simultaneamente, o canal de comunicação deveria ser capaz de transmitir uma informação a múltiplos destinatários. Além disso, pelo fato da aplicação ter como veículo alvo VANTs de pequeno porte, como apresentados anteriormente, a proposta deveria ser uma solução fisicamente leve e também pequena, a fim de não prejudicar o desempenho do veículo. Finalmente, o equipamento escolhido precisaria ser comercialmente viável. Para atender a esses requisitos optou-se pelos módulos *XBee - ZigBee Series 2* da empresa *Digi* a fim de servirem como link de comunicação entra a estação de controle e os VANTs. A Figura 3.3 mostra uma foto de um módulo deste modelo. Já a Tabela 3.3 resume as principais especificações técnicas desses dispositivos.

Figura 3.3 – Módulo XBee - ZB Series 2 utilizado.

Fonte: (XBEE)

Tabela 3.3 – Especificações Técnicas do Módulo XBee - ZB Series 2

TX Peak Current	40mA
RX Current	40 mA (@3.3 V)
Power-down Current	< 1 uA
Indoor/Urban	up to 40m (133 ft)
Outdoor line-of-sight	up to 120m (400 ft)
Transmit Power	2 mW (3 dBm)
Receiver Sensitivity	-96 dBm
Dimensions	24mm x 28mm x 9mm
Weight	3.24g

Fonte: (XBEE)

3.3.1 XBee

Os módulos *XBee* fazem parte de uma família de rádio frequência todos compatíveis entre si em relação a sua dimensão. Já no mercado desde 2005, suas primeiras versões possibilitavam apenas comunicação ponto-a-ponto. Os módulos adotados para o projeto possibilitam, além da configuração de redes ponto-a-ponto, a definição de topologias *point-to-multiple*, P2P e também rede *mesh* (rede de malha).

Em relação à forma, módulos *XBee* estão disponíveis em em dois formatos: o formato *through-hole* e o formato de montagem superficial (SMT). A versão utilizada neste trabalho foi no formato *through-hole*. Todos os dispositivos neste padrão estão disponíveis na configuração *20-pin*.

Em relação ao tipo de dados trocados na comunicação, cada módulo *XBee* pode ser operado de duas maneiras dependendo de seu modo de configuração: se no modo transparente (AT) ou no modo baseado em pacotes (API). No modo AT, dados entrantes no módulo são diretamente transmitidos pelo ar para o módulo destino, sem alterações nenhuma no formato dos dados. No modo API, por outro lado, os dados transmitidos são contidos em pacotes estruturados, os quais permitem alterações na forma de endereçamento, configuração de parâmetros e *feedback* de pacotes entregues.

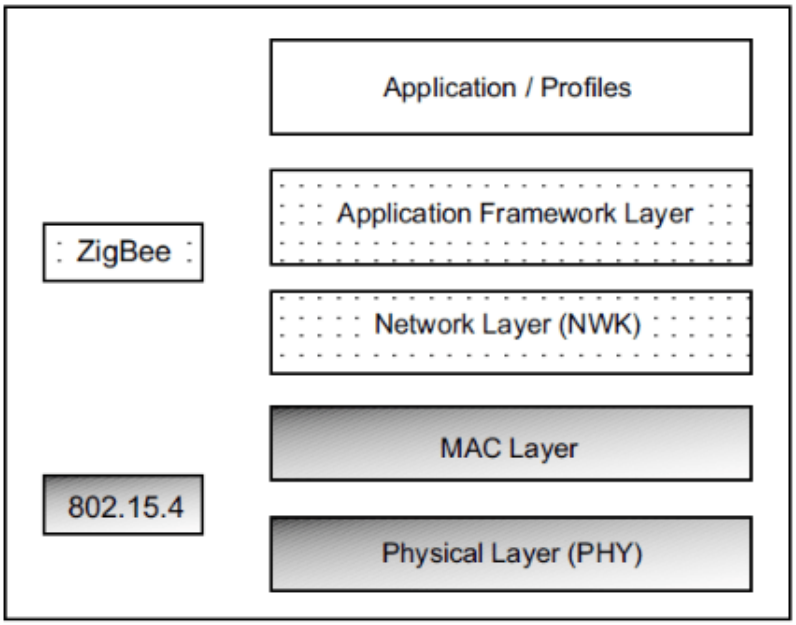
3.3.2 ZigBee

O termo *ZigBee* corresponde a um conjunto de especificações para protocolos de comunicação de *Personal Area Networks* (PANs) focadas em módulos de rádios pequenos com consumo baixo de energia. Esse conjunto de especificações foi criado com o intuito de ser menos custoso e mais simples do que outros protocolos como Wi-Fi e Bluetooth,

dado tipo de aplicação em que são usados.

A Figura 3.4 ilustra a disposição das camadas na arquitetura *ZigBee*. Nesta pilha, acima da especificação 802.15.4 são definidas camadas específicas para um elemento da rede *ZigBee*. A camada de rede provê as operações de roteamento, o que permite que pacotes passem por outros módulos *XBee* a fim de atingirem o destino final. As duas últimas camadas são responsáveis por dar suporte à aplicação, incluindo a descoberta de serviços e dispositivos oferecidos bem como recursos para gerenciamento da rede.

Figura 3.4 – Arquitetura de camada do protocolo ZigBee



Fonte: (DIGI)

Uma rede *ZigBee* consiste em três tipos diferentes de dispositivos: coordenador (*coordinator*), roteador (*router*) e dispositivos finais (*end-devices*), organizados em uma topologia semelhante à Figura 3.5. Todos esses tipos de elementos têm um papel distinto na rede criada e serão detalhados na sequência bem como o mecanismo de transmissão, roteamento e endereçamento entre estes.

3.3.2.1 Coordenador

O módulo coordenador é uma espécie de líder da rede, possuindo mais recursos em comparação aos outros três. Através das operações do coordenador, é que muitas das decisões são tomadas em relação aos outros módulos.

Cada rede possui exatamente um coordenador que deve estar sempre disponível,

ou seja, não pode estar em estado ocioso ou "dormindo"(*sleeping*). Além de estabelecer a rede, o coordenador tem as seguintes funções:

- Define o código da rede, necessário para outros módulos se juntarem.
- Permite ou não dispositivos roteadores e dispositivos finais de se juntarem a rede.
- Auxilia no roteamento.
- *Bufferiza* pacotes para serem entregues a dispositivos finais que se encontrem inativos no momento.

3.3.2.2 Roteadores

Roteadores atuam como nós intermediários, retransmitindo dados para outros nós. Assim como o coordenador, não pode estar ocioso, visto que também armazenam pacotes a serem retransmitidos para dispositivos finais. Roteadores são normalmente usados para estender a rede, visto que possuem a capacidade de repassar informação. As principais características dos roteadores podem ser resumidas da seguinte maneira:

- Deve se juntar a uma rede *ZigBee* antes de poder transmitir, receber e encaminhar dados.
- Após se juntar, pode auxiliar o coordenador na função de permitir que outros roteadores e dispositivos finais sejam adicionados à rede.
- Como o próprio nome sugere, realizam o roteamento das informações.
- Também podem, assim como o coordenador, armazenar pacotes para dispositivos finais.

3.3.2.3 Dispositivos finais

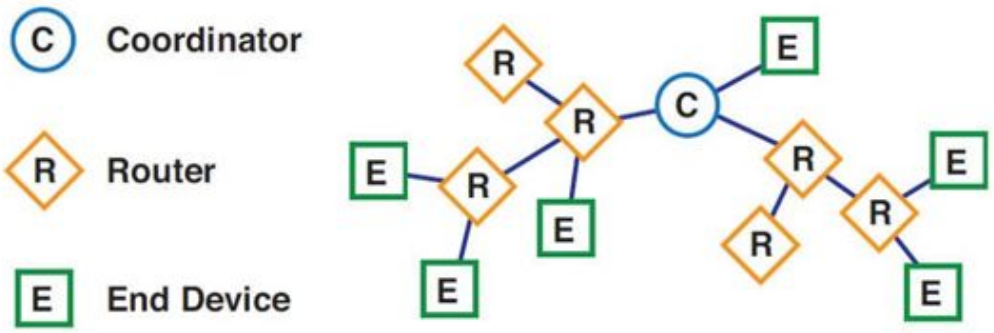
A classificação de dispositivos finais geralmente está associada a equipamentos de baixo consumo de energia como sensores, por exemplo. Esses dispositivos possuem funcionalidade suficiente para comunicação ou com um roteador ou com o coordenador, não podendo encaminhar dados para nenhum outro módulo. Essa operação reduzida permite economizar consideravelmente o uso de energia.

Ao entrar em uma rede, o módulo resposável por autorizar esse procedimento se torna o nó pai desse dispositivo final. Resumidamente, os *end-devices* possuem as seguintes características:

- Não podem permitir a entrada de novos dispositivos na rede.

- Sempre transmitem dados apenas para o módulo pai.
- Não encaminham dados.
- Podem entrar em estado ocioso a fim de economizar energia.

Figura 3.5 – Topologia de uma rede ZigBee



Fonte: (DIGI, 2014)

3.3.2.4 Endereçamento e transmissão

Todos os dispositivos *ZigBee* possuem dois endereços diferentes: um de 64 bits e outro de 16 bits. O primeiro endereço corresponde a um valor único e fixo para cada módulo, atribuído no momento de sua fabricação. Já o endereço de 16 bits corresponde ao endereço atribuído ao dispositivo ao se conectar a uma rede *ZigBee*.

Todas as transmissões são realizadas utilizando o endereço de 16 bits. Entretanto, esse valor não é estático, pois pode mudar em duas situações: a primeira caso haja um conflito entre dois módulos com o mesmo número e a segunda caso um dispositivo saia da rede, entrando novamente após um tempo, recebendo um identificador diferente. Por essa razão, o endereço de 64 bits é geralmente incluso durante as transmissões, garantindo a entrega para o destino correto em casos de conflitos ou invalidação desse endereço de 16 bits.

Em relação à transmissão, pacotes podem ser entregues em *unicast* ou em *broadcast*. Transmissões em *broadcast* no protocolo *ZigBee* têm a intenção de serem transmitidas para todos os nós da rede. Para evitar que mensagens sejam replicadas de forma infinita, cada nó possui uma tabela contendo cada mensagem recebida via *broadcast*. As transmissões em *unicast* podem ser destinadas a um vizinho imediato ou para outro destinatário em qualquer local da rede, necessitando, dessa forma de soluções de roteamento.

A Tabela 3.4 apresenta os diferentes tipos de soluções de roteamento adotadas pelos dispositivos.

Tabela 3.4 – Abordagens adotadas para roteamento

Abordagem	*Descrição*	*Aplicação*
Ad-hoc On-deman Distance Vector Mesh Routing	Algoritmo de vetor de distâncias em que cada dispositivo sabe quem é o próximo nodo no caminho até o destino.	Usada em redes que não passem de 40 dispostivos.
Roteamento *Many-to-one*	Única transmissão em *broadcast* configura rotas em todos os dispositivos.	Adequada quando múltiplos dispostivos remotos enviam dados para um único dispositivo.
Roteamento por origem	Pacote e dados contém a rota completa até o destino.	Melhora eficiência do roteamento para redes com mais de 40 dispositivos.

Fonte: (DIGI, 2015)

3.3.3 Configuração dos Módulos XBee

A fim de configurar os módulos *XBee* corretamente, foi preciso compreender os requisitos que a rede criada deveria oferecer. Um elemento nessa rede deveria ser capaz de transmitir mensagens a todos os outros dispositivos, ao passo que esses outros dispositivos deveriam ser capaz de responder a esse elemento. Traduzindo para o contexto deste projeto, a estação de controle precisaria conversar com todos os VANTs e estes exclusivamente com a estação de controle. Dessa forma, optou-se por uma modelagem *point-to-multiple*.

A configuração dos módulos se deu utilizando o *software* X-CTU (XCTU), plataforma oficial da Digi desenvolvida principalmente para a configuração do *firmware* dos módulos. X-CTU é uma ferramenta gratuita e multiplataforma (disponível em ambientes *Linux*, *Windows* e *MacOS*) desenvolvida com o intuito de permitir que usuários interajam com os módulos *XBee* através de uma interface gráfica. Além disso, conta com diversos recursos interessantes como a visualização da topologia da rede criada, testes de comunicação dos módulos configurados, entre outros.

3.3.3.1 Geração da Rede

A fim de gerar a rede, inicialmente foi preciso definir um módulo para servir como coordenador, pré-requisito fundamental para se criar uma rede *ZigBee* como discutido anteriormente. A principal característica que esse dispositivo deveria ter era poder se comunicar com todos os outros módulos da rede. Para isso, foi utilizada a função de *broadcast*, configurando o endereço de destino do módulo para tal função.

Para os VANTs optou-se por usar módulos com função de roteador, visto que estes não entram em modo ocioso, como os dispositivos finais, mas sim permanecem ativos o tempo todo. Como a troca de mensagens é constante entre os VANTs e a estação de controle, os módulos tem de ficar constantemente operantes para poder suportar o volume de mensagens MAVLink trocadas.

Na configuração desta rede, cada VANT comunica-se exclusivamente com o coordenador em *unicast*. Para isso, cada módulo teve programado no seu *firmware* o endereço do módulo do coordenador.

Com isso, foi possível gerar uma rede como a ilustrada na Figura 3.6. A Figura 3.7 ilustra a conexão dos módulos no dispositivo Android e no computador conectados ao simulador.

Figura 3.6 – Comunicação *point-to-multiple* de módulos *XBee*

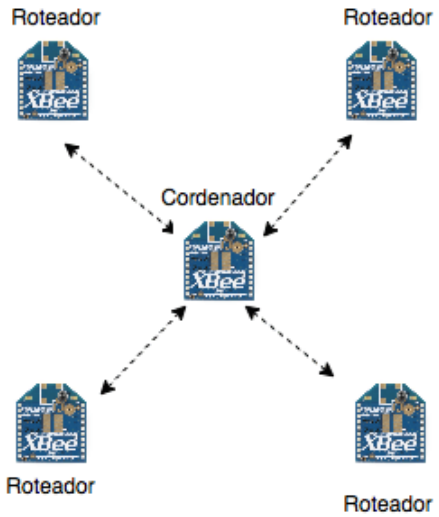
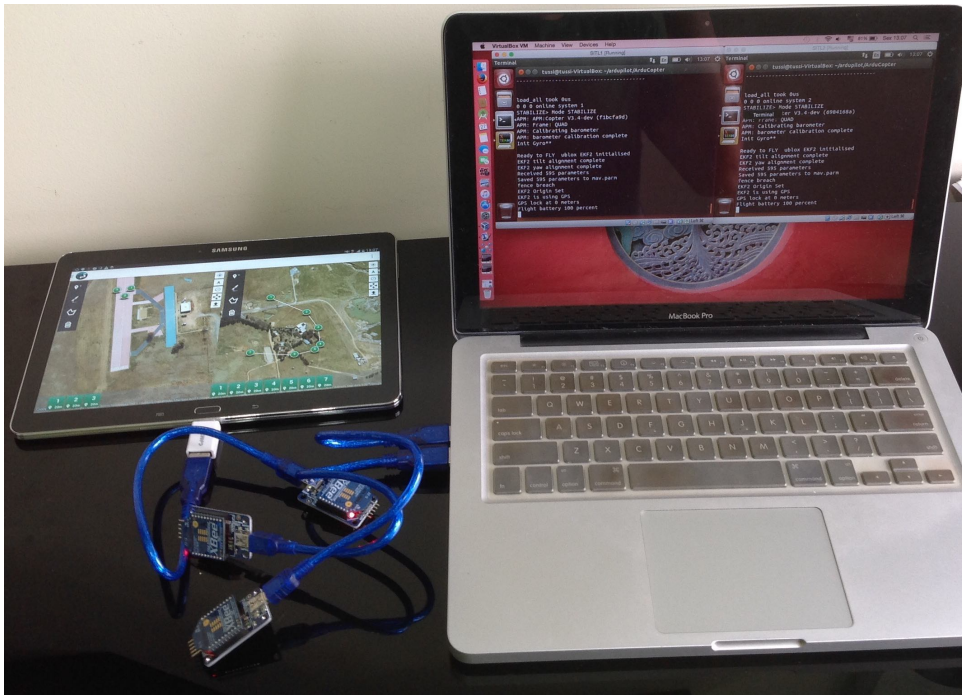


Figura 3.7 – Conexão dos módulos ao tablet e ao computador



4 IMPLEMENTAÇÃO

O aplicativo, relativo a sua implementação, foi concebido com o auxílio da IDE *Android Studio*. Essa plataforma foi anunciada pela Google em 2013 para ser usada como plataforma oficial para desenvolvimento de projetos Android. Sua primeira versão estável foi liberada para uso em 2014 sob licença Apache, estando atualmente na versão 1.5.

A fim de tornar o processo de desenvolvimento mais organizado, foi usado o sistema de controle de versão Github. Esse serviço, além de manter o histórico das alterações no código, facilita na sua distribuição e permite a participação colaboradores.

Este capítulo discorrerá inicialmente, de uma forma mais geral, acerca das etapas de desenvolvimento do projeto. Na sequência, serão apresentadas em maiores detalhes as modificações na arquitetura e no fluxo geral do sistema. Finalmente, serão abordadas as mudanças na interface bem como a justificativa das escolhas para as mesmas, além das novas funcionalidades.

4.1 Etapas

A extensão do aplicativo DroidPlanner passou por diferentes etapas até a concepção da plataforma no seu estado atual. Inicialmente, foi necessário compreender os principais elementos existentes no trabalho original, além da comunicação entre os mesmos. A boa compreensão do sistema foi crucial para as decisões tomadas no que diz respeito aos componentes que deveriam ser modificados para a solução proposta. Na sequência, foram feitas as primeiras modificações relativas à comunicação e às estruturas internas, incluindo a definição de nova mensagem MAVLink. Finalmente, tendo como base as modificações das estruturas e levando em consideração as exigências da nova interface, foram realizadas as alterações na parte da interação com o usuário.

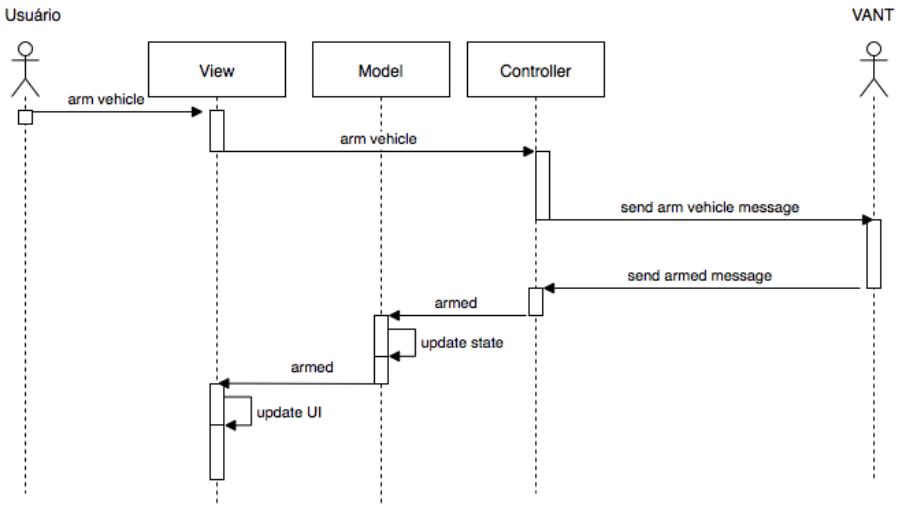
4.1.1 Familiarização com os componentes preexistentes

Antes de começar a implementação, foi necessário entender o funcionamento do DroidPlanner a partir da sua arquitetura, Model-View-Controller (MVC). Conforme detalhado no projeto original (BENEMANN, 2013), a parte *model* corresponde à representação interna do VANT, ou seja, seu estado como visto pela plataforma de controle. A

parte *view* representa à parte com a qual o usuário interage, ou seja, a interface gráfica. Por fim, a parte *controller* corresponde ao recebimento de tráfego do VANT através das mensagens MAVLink, traduzindo-as e as transformando em atualizações tanto no modelo quanto na interface.

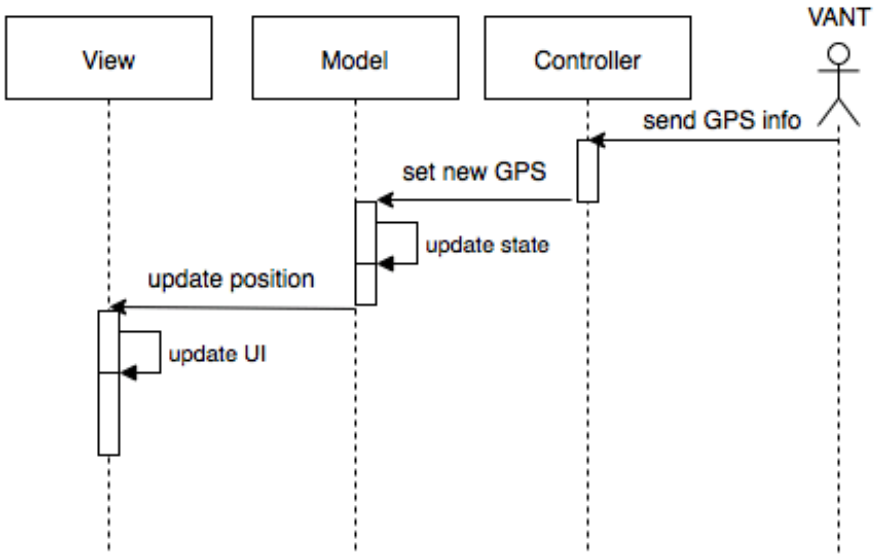
O diagrama de sequência simplificado e de alto nível da Figura 4.1 ilustra um cenário de interação entre as três visões da arquitetura. Neste cenário, usuário comunica-se com a interface (*View*), enviando um comando ao VANT. O sistema então, prepara a mensagem adequada e a envia para o VANT através do canal de comunicação sendo utilizado. Ao receber essa mensagem, VANT a processa, informando seu estado modificado através de uma outra mensagem de volta para a estação de controle. Finalmente, DroidPlanner, ao receber essa informação, atualiza seu estado interno, gerando notificações para a representação interna do VANT (*model*) e este, por sua vez, notificando a interface.

Figura 4.1 – Diagrama de sequência de um cenário onde usuário envia um comando.



Nem todas as mensagens recebidas pelo VANT são consequência de um comando direto do usuário. Um outro cenário possível, ilustrado na Figura 4.2, mostra o que ocorre com as mensagens de estado enviadas pelo VANT periodicamente sem serem transmitidas como consequência da ação do usuário. Neste caso, assim como anteriormente, notificações são geradas a fim de atualizar os elementos *model* e *view*.

Figura 4.2 – Diagrama de sequência representando cenário com mensagens enviadas pelo VANT



4.1.2 Link de Comunicação

Inicialmente, principalmente para as fases inicias dos testes, foi definida a utilização do protocolo UDP para comunicação com as múltiplas instâncias dos VANTs, visto ser uma alternativa menos pesada ao TCP, dados seus controles de fluxo e de congestionamento, que seriam penosos para o número de mensagens trocadas proporcional ao aumento do número de veículos sendo controlados. Por outro lado, a perda de pacotes ocasionada pela falta de mecanismos de entrega confiável de mensagens, fez com que comandos tivessem de ser reenviados pelo usuário em alguns momentos, situação observada conforme aumentou-se o número de VANTs sendo controlados simultaneamente.

Para enviar as mensagens para os VANTs, utilizou o mecanismo de *broadcast* oferecido pelo protocolo UDP, simplificando o envio de mensagens da plataforma para o simulador. Apesar de todos os VANTs receberem as mensagens, estes apenas as processam efetivamente caso conteúdo da mensagem MAVLink estiver endereçado com seu identificador destino único.

A segunda abordagem para estabelecer o link de comunicação foi a utilização dos módulos *XBee*. Entretanto, a nível de implementação, não foram necessárias mudanças no código, visto que foi utilizada a interface USB para saída e entrada de dados. A modificação, de fato, se deu na configuração do *firmware* dos módulos *XBee*, como já discutido anteriormente.

4.1.3 Estruturas Internas

As modificações das estruturas internas foram necessárias em todos aqueles elementos que manipulavam apenas um VANT no aplicativo, tanto em relação a sua modelagem interna, como suas variáveis representando seu estado, quanto aos métodos invocados, visto que agora a identificação dos VANT deveria ser considerada a fim de diferenciar suas ações no aplicativo. Nesta seção, o termo VANT se refere aos componentes do sistema que modelam o VANT físico, ou seja, as classes, atributos e métodos que representam o veículo sendo virtualizado.

A principal classe do sistema, cujas modificações nas estruturas foram base para o resto da aplicação, foi a classe *DroidPlannerApp*. Através dessas alterações foi possível definir o funcionamento do resto da aplicação, visto que esse módulo é responsável por fazer a interface entre a parte do recebimento de mensagens com a atualização do resto do sistema.

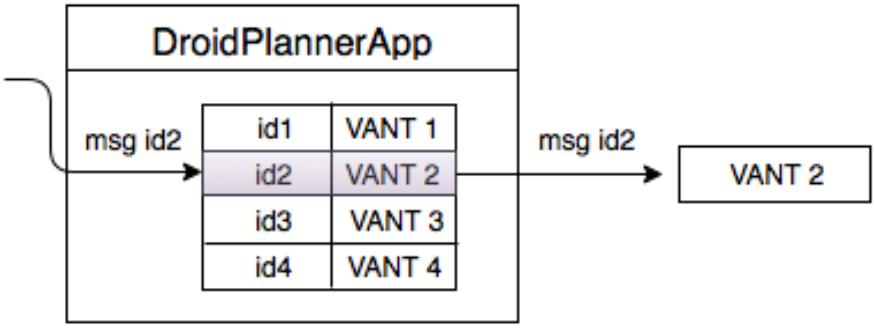
DroidPlannerApp é uma classe base que armazena o estado global da aplicação, mantendo informações como o VANT modelado e as missões criadas. Além disso, é responsável por receber novas mensagens oriundas das *threads* monitorando o canal de comunicação, repassando-as para diversos outros elementos do sistema.

A primeira mudança nessa classe foi a criação de uma estrutura que pudesse armazenar um conjunto de VANTs. Para isso, criou-se uma tabela *hash* cuja chave de acesso correspondente é o código do VANT, valor único informado pelas mensagens recebidas. Sempre que qualquer elemento do sistema necessitasse consultar informações de um veículo específico, bastava consultar as informações contidas nessa tabela. Em relação à população dessa estrutura, sempre que um novo VANT era detectado, através do identificador deste como emissor das mensagens recebidas, era gerada uma nova entrada na tabela.

Outra mudança vital foi a entrega de mensagens recebidas de acordo com o VANT destino. A fim de manter o sistema consistente era necessário garantir que as mensagens fossem entregues para a instância do VANT correto. A Figura 4.3 ilustra um cenário em que o identificador do emissor da mensagem é analisado para repassá-la para o modelo correto.

As demais alterações no código em relação às estruturas internas tiveram como base consultas à tabela *hash* dos VANTs.

Figura 4.3 – Consulta à tabela hash ao recebimento de uma mensagem



4.1.4 Definição de nova mensagem MAVLink

A fim de implementar algumas funcionalidades extras, foi necessário definir uma nova mensagem MAVLink, contendo informações personalizadas para a comunicação da estação de controle com os veículos.

A mensagem foi definida inicialmente em XML, sendo posteriormente traduzidas para código Java. Em relação ao seu conteúdo, este foi projetado para conter informações referentes ao tipo de padrão de reconhecimento a ser utilizado pelos VANTs para cobrir uma área definida no mapa. Seu formato em XML está representando no Quadro 4.1.

Quadro 4.1 – Definição da nova mensagem MAVLink criada.

```
<message id="200" name="RECOGNITION_PATTERNS">
        <description>Message contains which recognition patter will be used</description>
        <field type="uint8_t" name="target_system"> System ID (0 for broadcast)</field>
        <field type="uint8_t" name="target_component"> Component ID (o for broadcast) </
            field>
        <field type="uint32_t" name="recognition_pattern"> Contains which recognition
            patter will be used (255 patterns possible) </field>
    </message>
```

Como dito anteriormente na seção 2.2.2, para a identificação desta mensagem, foi usado o intervalo reservado entre 150 e 240. O campo *target_system* indica para qual VANT essa mensagem está sendo entregue. O valor *target_component* é usado para caso a mensagem seja destinada a um componente específico do VANT identificado por *target_system*. Finalmente, campo *recognition_pattern* contém o tipo de padrão de reconhecimento dos elementos da missão que deve ser utilizado.

4.1.5 Modificações na Interface

Finalmente, a última etapa consistiu em realizar as modificações na interface, levando em consideração o design original bem como os novos elementos importantes para a visualização e manipulação simultânea dos VANTs sendo controlados. As mudanças na interface foram uma parte crucial para o projeto, visto que é a parte com a qual o usuário final interage, além de ser o reflexo das mudanças internas. A seção 4.3 discorrerá acerca dos elementos adicionados à interface.

4.2 Arquitetura

O projeto manteve a arquitetura baseada no modelo MVC, visto esta ainda ser uma boa opção de organização para o projeto atual por permitir que os diferentes módulos do sistema possam ser implementados de forma independente, facilitando a integração dos elementos. Entretanto, algumas mudanças na arquitetura original MVC tiveram de ser modificadas para representar o sistema corretamente.

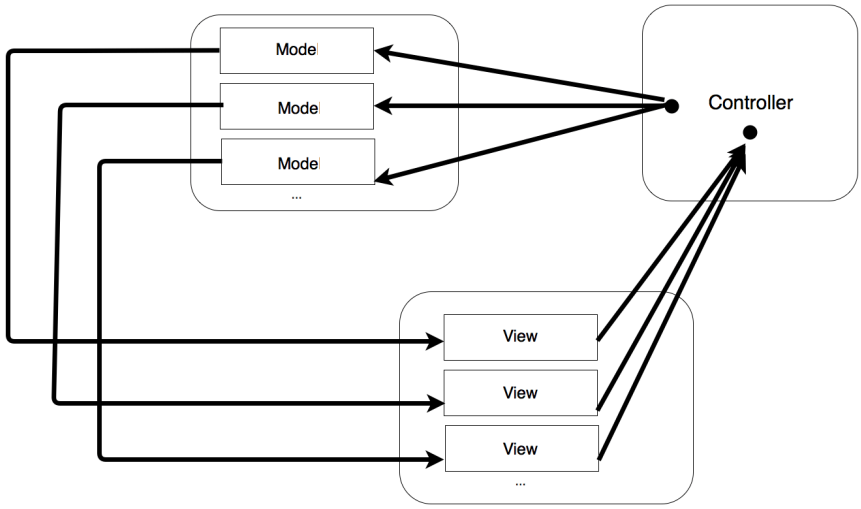
A Figura 4.4 ilustra, através de uma diagrama, a arquitetura adotada, bem como comunicação entre os três componentes do modelo MVC. Neste diagrama, a parte de controle representa tanto o recebimento de tráfego, quanto o envio de dados da plataforma para o VANT. A visão, corresponde às múltiplas instâncias da interface original na solução atual. Finalmente, a parte de modelo, representa a virtualização dos diversos VANTs físicos sendo controlados no sistema.

4.2.1 Controle

A parte de controle é responsável por gerenciar todo o recebimento de mensagens MAVLink, realizando a decodificação destas e as encaminhando para a representação interna do VANT correto. Além de tratar mensagens recebidas, esse componente também prepara mensagens para serem enviadas para o VANT físico através do link de comunicação estabelecido.

O bloco de controle também se comunica com uma interface responsável por estabelecer o canal de comunicação com o VANT físico. Os protocolos de comunicação implementados neste trabalho são o UDP e a comunicação utilizando os módulos *XBee*

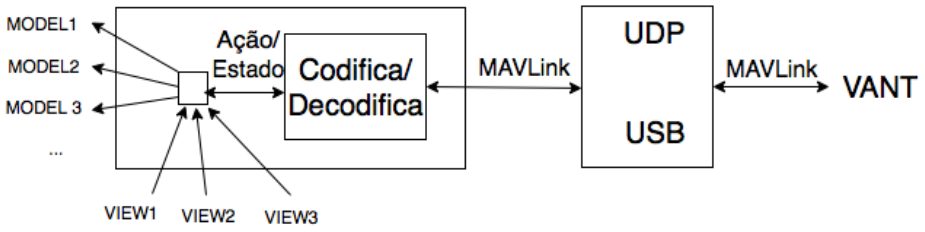
Figura 4.4 – Diagrama representando a arquitetura MVC modificada



que se comunicam com a plataforma de controle através de um canal serial USB. A Figura 4.5 ilustra através de diagramas todas essas interações.

A principal mudança necessária no controle foi a separação das mensagens recebidas para a entrega correta ao modelo virtualizado correspondente ao VANT emissor da informação. Já no envio de mensagens da estação de controle, todas essas são agregadas e enviadas de maneira sequencial conforme forem chegando da interface.

Figura 4.5 – Diagrama representando o módulo de controle



4.2.2 Modelo

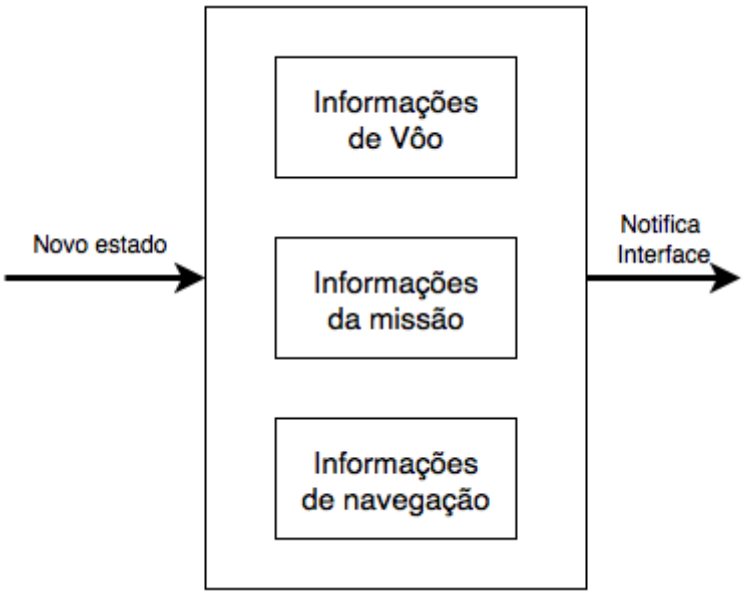
O componente de modelo, como dito anteriormente, é resposável pela virtualização do VANT físico no sistema, guardando e modificando seu estado a partir das mensagens MAVLink recebidas. Além de armazenar informações sobre os VANTs, esse com-

ponente tem a importante função de notificar a interface das alterações ocorridas no seu estado.

Neste trabalho, o modelo teve de ser modificado para poder controlar múltiplos veículos simultaneamente. Para isso, esse componente da arquitetura MVC foi replicado, em contraste com o projeto original, em vários outros elementos estruturalmente idênticos, porém contendo informações distintas por virtualizarem veículos diferentes.

A Figura 4.6 mostra uma visão interna de cada bloco do modelo. Todas as mensagens recebidas pela estação de controle e destinadas ao VANT correspondente são processadas por esse modelo, alterando os componentes necessários para, finalmente, notificar a interface das mudanças realizadas.

Figura 4.6 – Visão interna do modelo de representação de um VANT



4.2.3 Visão

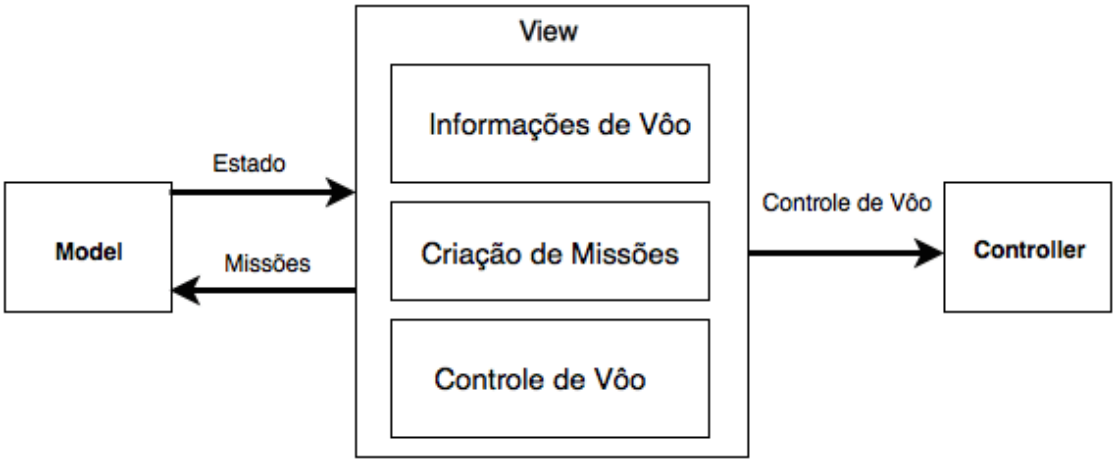
A visão do componente MVC representa a interface com usuário. Esta reflete todas as mudanças ocorridas no modelo bem como recebe todas as ações do usuário para controlar os múltiplos VANTs.

Apesar da interface com o usuário ser apenas uma, correspondendo à tela com a qual o humano interage, para tornar mais claro o entendimento das mudanças na parte da visão, este componente foi também replicado na representação da arquitetura em várias

instâncias de acordo com o número de VANTs sendo controlado, visto que cada instância corresponde a uma parte da interface final independente das demais.

Cada elemento da visão representa um pedaço da interface, responsável por atualizar informações de um dos veículos sendo controlado, bem como por receber ações do usuário para o controle deste veículo. A Figura 4.7 ilustra os principais itens presentes em cada componente da visão, bem como a interação com os outros elementos da arquitetura.

Figura 4.7 – Arquitetura da interface com o usuário para um VANT



4.3 Interface com Usuário

O controle de múltiplos veículos simultaneamente através de uma plataforma de controla faz com que surjam novos desafios não só na parte de comunicação, mas também na parte de interação com o usuário. O controle e a visualização dos VANTs de forma dinâmica são duas das principais funcionalidades que uma estação de controle deve prover de forma adequada a fim de que seja factível a manipulação e monitoração desses veículos em tempo real.

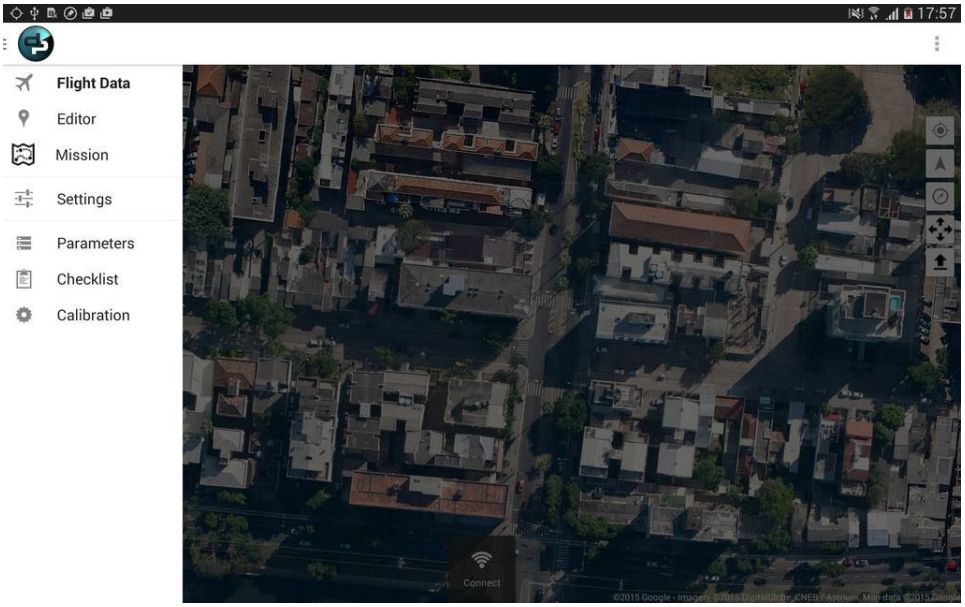
Levando em consideração essas questões, foi possível compreender os principais requisitos e funcionalidades que a aplicação deveria prover. Abaixo estão listadas as principais necessidades detectadas:

- Usuário deve ser capaz de visualizar, na mesma tela, todos os VANTs ativos sendo controlados simultaneamente.
- Deve ser possível comandar individualmente cada veículo com a mesma simplicidade da versão original.

- As informações de cada veículo ativo devem aparecer simultaneamente para o usuário a fim de que este possa monitorar o *status* de cada VANT.
- Deve ser permitido ao usuário mudar o modo de visualização de acordo com a necessidade.

Nesta seção serão apresentadas todas as modificações realizadas na interface a fim de acomodar todas essas funcionalidades necessárias para o controle simultâneo de múltiplos veículos. Para isso serão apresentados os principais contextos da aplicação, responsáveis por implementar todos esses recursos. A navegação entre esses contextos é realizada através de um menu de navegação, como ilustrado na Figura 4.8. As principais modificações ocorreram no contexto de comando e visualização (*Flight Data*), contexto de planejamento de missões (*Editor*) e contexto de visualização da missão (*Mission*).

Figura 4.8 – Configuração do Novo Menu de Atividades



4.3.1 Comando e Visualização

O contexto de comando e visualização é responsável pelo acompanhamento e controle dos VANTs sendo manipulados, duas das funções mais cruciais de uma estação de controle. Por causa disso, uma série de mudanças foi realizada na interface a fim de facilitar o controle para o usuário. Inicialmente, foi preciso fragmentar a tela em um número de pedaços proporcional ao número de VANTs conectados. Na sequência, todas as infor-

mações e mecanismos de controle tiveram de ser separados e individualizados para cada veículo.

4.3.1.1 Múltiplas telas de Visualização

O mecanismo básico de visualização adotado para a extensão do DroidPlanner foi a criação de múltiplos fragmentos de visualização da interface, onde cada parte contém informações de um dos VANTs sendo controlados. O principal motivo pelo qual essa solução foi adotada foi, como já listado, a necessidade de monitorar simultaneamente todos os dispositivos sendo controlados, tendo uma visão global da situação de cada um.

A Figura 4.9 ilustra a disposição de alguns dos elementos, durante a fase de implantação, de um cenário hipotético da divisão da interface em quatro partes. Cada elemento, como é possível observar, possui seus próprios botões de interação, próprios painéis de informação de voo e, principalmente, um mapa, sobre o qual todas os elementos estão dispostos.

O mecanismo de instanciação desses elementos foi implementando levando-se em consideração a conexão dinâmica de novos VANTs. Dessa forma, ao se iniciar o aplicativo, temos apenas uma tela de visualização, esta se dividindo conforme aplicação percebe a conexão de novos veículos.

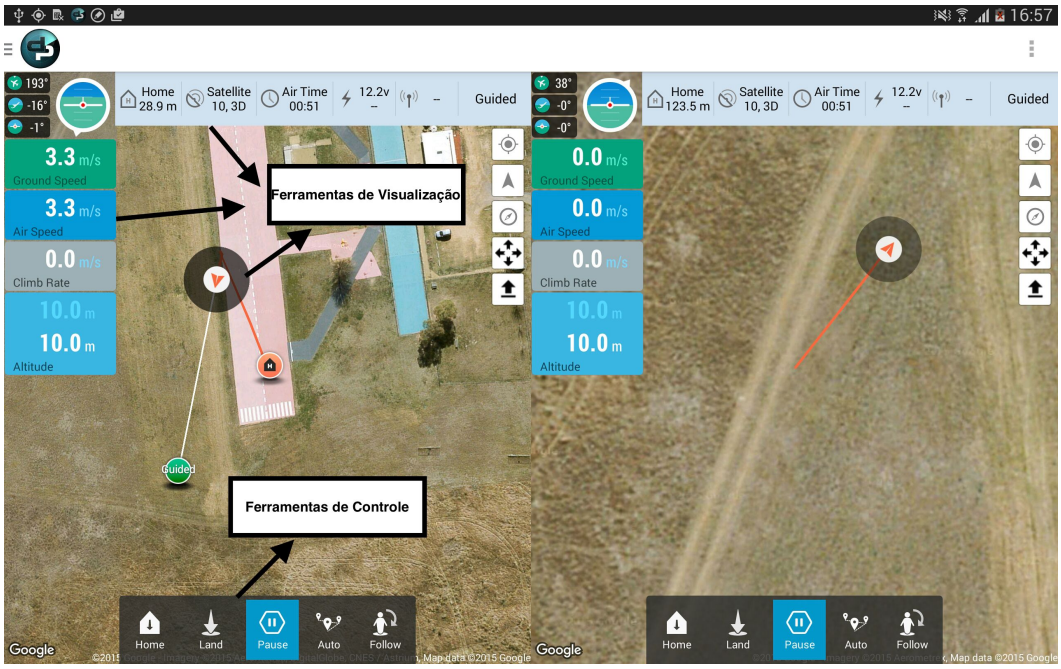
Figura 4.9 – Múltiplas telas de visualização

4.3.1.2 Controle e visualização de informação dos VANTs

O controle de VANTs no que diz respeito à interface, se refere a todos os itens presentes na tela com os quais o usuário interage e que se traduzem em comandos aos veículos físicos. A parte de visualização, por outro lado, corresponde às informações a respeito do estado do VANT no momento, tanto em relação à posição de um veículo, quanto sua altitude, velocidade e demais informações relevantes.

A Figura 4.10 ilustra os elementos de controle e de visualização da interface em um cenário em que dois veículos estão sendo controlados simultaneamente. Seguindo o princípio de múltiplas telas, cada fragmento da interface representando um VANT ativo possui seus próprios elementos de controle e de visualização.

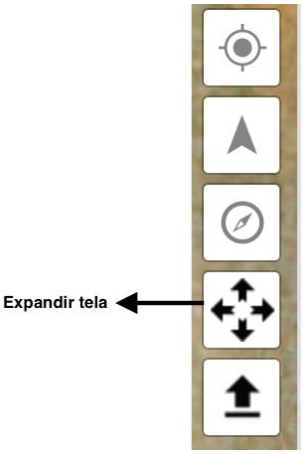
Figura 4.10 – Controle e visualização simultânea de 2 VANTs



Percebendo o aumento no número de informações exibidas para o usuário conforme aumenta-se o número de veículos sendo controlados, gerando uma interface carregada visualmente, fez-se necessário incluir uma nova funcionalidade para expandir um fragmento de tela de um VANT escolhido. Para isso, foi preciso acrescentar um ícone que permitisse esse recurso.

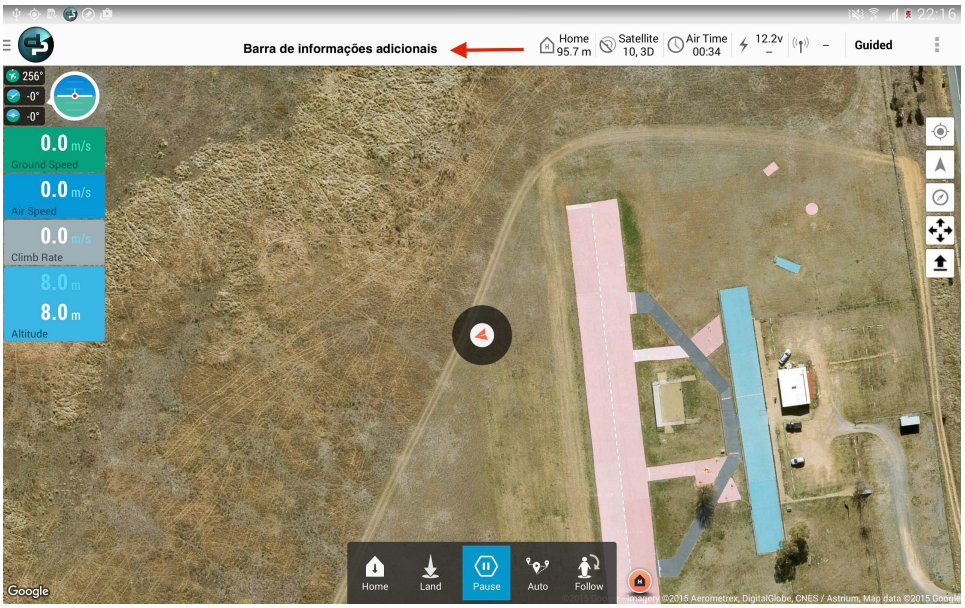
A Figura 4.11 ilustra o botão responsável por entrar em modo tela cheia. Ao ser clicado, a visão do usuário é alterada, fazendo com que apenas as informações do VANT da tela escolhida sejam visíveis. A Figura 4.12 mostra o resultado da tela expandida. Para

Figura 4.11 – Novo conjunto de botões adicionado



o modo tela cheia, optou-se por mover a barra de informações adicionais para o topo da tela, a fim de deixar a visualização do mapa mais abrangente.

Figura 4.12 – Tela expandida com foco em um único VANT

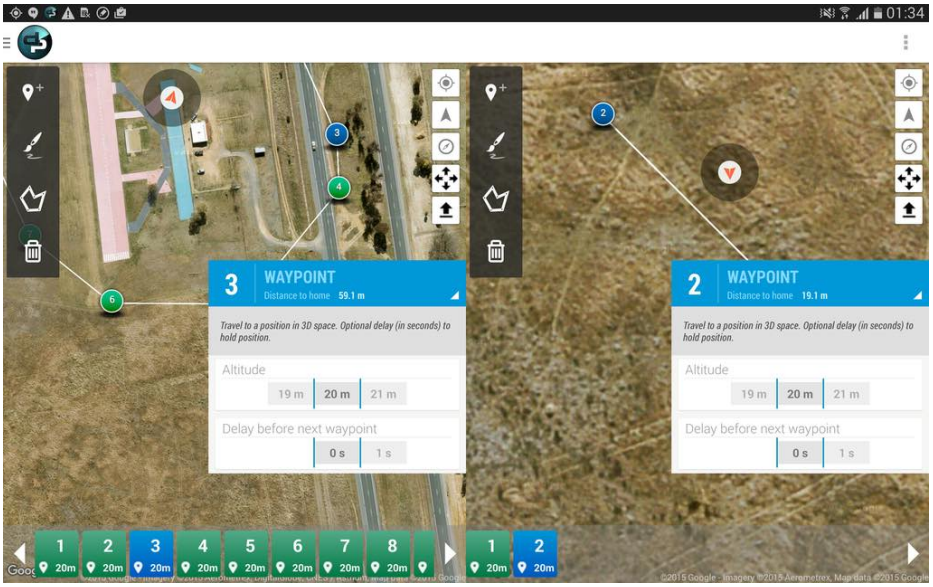


4.3.2 Planejamento de Missões

O contexto de planejamento de missões se refere às atividades que permitem ao usuário definir pontos de interesse (*waypoints*) por onde o VANT deve passar. Cada ponto possui informações como altura, tipo de ação a ser executada sobre aquele local como, por exemplo, pousar ou decolar, entre outros dados relevantes que podem ser configurados. A

Figura 4.13 ilustra esse mecanismo.

Figura 4.13 – Configurando Waypoints

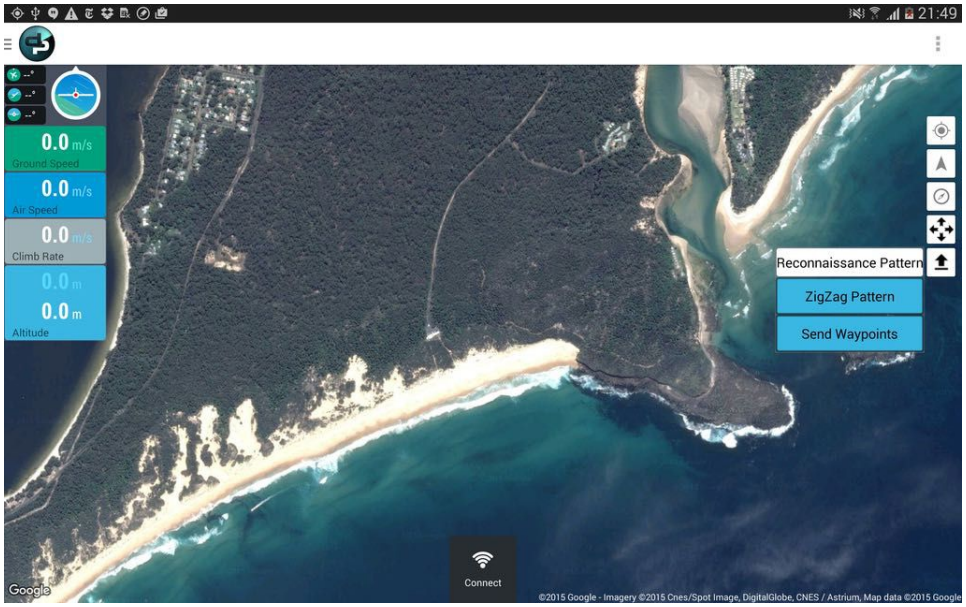


Para o planejamento de missões, foi necessário criar uma interface que também desse suporte a visualização múltiplos VANTs, permitindo que missões fossem criadas conforme o projeto original, mas simultaneamente para todos os veículos conectados. Para isso, assim como no contexto de controle e visualização, a interface foi alterada de modo que cada VANT conectado recebesse uma porção da tela. A Figura 4.14 ilustra o resultado no cenário de dois VANTs conectados.

Figura 4.14 – Criação de Missões

Além da fragmentação da tela, foi necessário adicionar um mecanismo de envio das missões criadas para os VANTs. Para isso, foi adicionado um botão lateral (como mostrado na Figura 4.11), junto aos ícones já existentes no DroidPlanner original a fim de prover essa funcionalidade. Esse botão, além de simplesmente enviar missões, oferece a possibilidade de indicar que tipo de algoritmo de reconhecimento de *waypoints*, implementado e executado pelo VANT físico, o veículo deve utilizar para realizar a missão. A Figura 4.15 ilustra um menu com algumas possibilidades.

Figura 4.15 – Menu dos Algoritmos de Reconhecimento



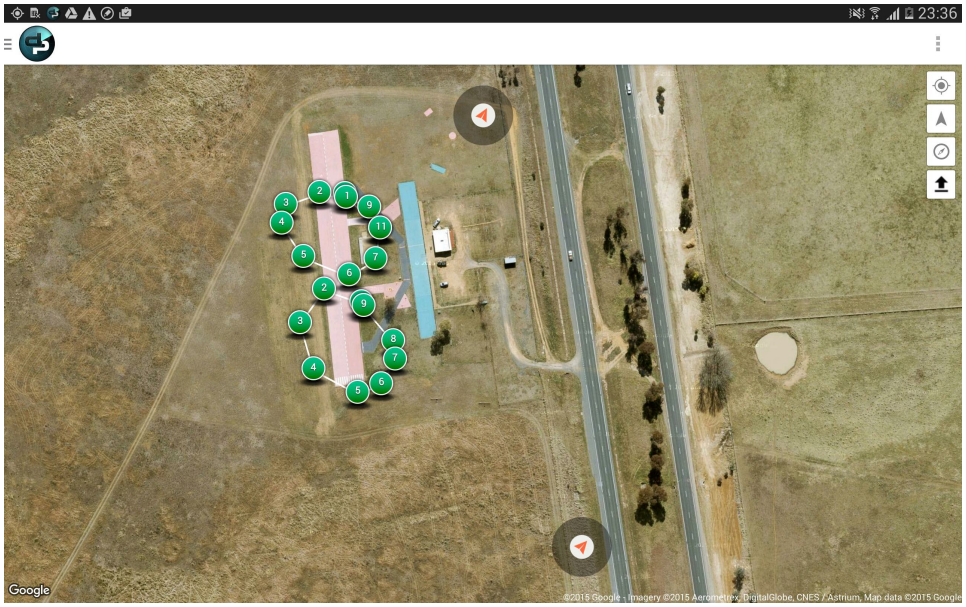
4.3.3 Visualização de Missões

O contexto de visualização de missões foi criado com o intuito de permitir ao usuário alternar a visualização individual e segmentada de cada veículo, para uma visão global do sistema em que todos os VANTs bem como seus *waypoints*, demarcando suas missões, estejam dispostos sobre o mesmo mapa. Por ser uma interface focada em visualização, apenas foram mantidos na tela os ícones essenciais para esse fim, mantendo o mapa o mais livre de elementos possível.

Em relação aos botões no lado direito do mapa, como é possível observar na Figura 4.16, foi retirado o ícone de expandir a tela, visto não ser um recurso necessário nesse contexto. Os demais botões foram mantidos, pois ainda são necessárias para o contexto.

Finalmente, o ícone de envio de missões, para esse contexto, foi configurado para

Figura 4.16 – Visualização das missões criadas



enviar todos os pontos criados para seus respectivos VANTs. Esse recurso foi programado dessa forma visando facilitar para o usuário a definição de missões para os múltiplos veículos, evitando a necessidade de se repetir o envio individualmente para cada VANT.

5 EXPERIMENTOS

Todos os experimentos realizados tiveram como ambiente o simulador descrito já apresentado anteriormente, visto ser este um ambiente seguro e de fácil acesso por não precisar utilizar nenhum *hardware* específico como veículos VANT, sistema de GPS e demais sensores. Para cada VANT controlado, foi preciso executar uma nova instância do simulador, configurando alguns parâmetros específicos para o veículo simulado como seu identificador, que deve ser diferente dos demais a fim de que a aplicação consiga diferencia-los.

Em relação ao número máximo de veículos controlados simultaneamente para as simulações, dois fatores foram limitantes: o primeiro deles se refere à disposição dos elementos da interface. Um número muito grande de informações referente aos vários VANTs faz com que não seja possível controlar adequadamente nenhum deles dado o espaço na tela. O segundo fator se refere à quantidade de mensagens recebidas. A cada novo veículo controlado, o número de mensagens dobra, fazendo com que o aplicativo da estação de controle tenha de processar muito mais informações por segundo.

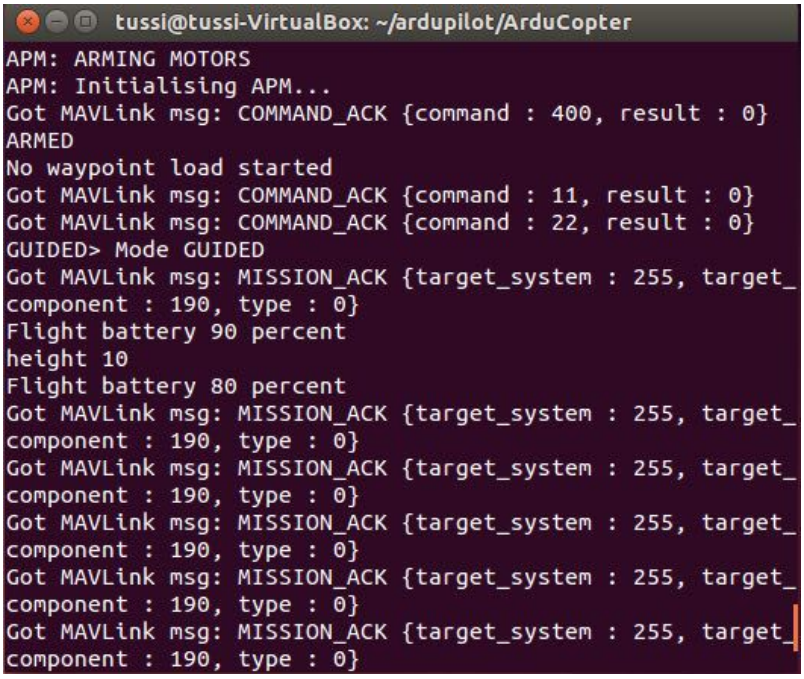
Analisando o recebimento de mensagens, percebe-se que o grande gargalo da aplicação é o fato de possuir apenas um canal de comunicação para tratar todas as mensagens recebidas. Mesmo que cada fragmento da interface seja atualizado de forma paralela, as mensagens acabam chegando de forma sequencial no meio físico, linearmente sobrecarregando o sistema conforme o número de veículos.

A Figura 5.1 mostra a captura de tela de uma instância do simulador utilizada. Todos os comandos enviados pela estação de controle e destinados para instância controlando um VANT são refletidas nessa tela, facilitando a depuração e confirmando a corretude dos comandos enviados.

As imagens a seguir ilustram o controle simultâneo de dois, três e quatro VANTs. Acima desse número, além de não ser possível organizar a tela de uma forma clara e funcional como as outras, deixaria a aplicação sobrecarregada em relação ao grande volume de informações recebido através do canal de comunicação.

O experimento iniciou-se com a abertura da conexão, fazendo com que a aplicação ficasse ouvindo o canal para o recebimento de mensagens MAVLink. A Figura 5.2 mostra a tela inicial bem como o botão de abertura da conexão. A adição de novos veículos ocorre de forma dinâmica, ou seja, conforme aplicação recebe mensagens de diferente VANTs (oriundas das diferentes instâncias do simulador), automaticamente atualiza a

Figura 5.1 – Interface do simulador utilizado



visão da interface para o usuário, mostrando todas as informações do novo VANT no seu fragmento de tela.

A Figura 5.3 ilustra o controle de dois VANTs no modo guiado, em que usuário define o ponto no mapa para onde o veículo deve se deslocar. Observou-se na prática que a divisão da tela em duas metades iguais foi suficiente para dispor todos os elementos e ainda assim obter uma boa porção do mapa livre para a visualização dos veículos.

A Figura 5.4 mostra o experimento da parte de planejamento de missões com mais um VANT conectado. A tela maior da direita, por razões óbvias, tornou a demarcação dos pontos a serem visitados mais fácil de ser estabelecida em comparação com as miniaturas da esquerda. Em relação ao desempenho do sistema como um todo, percebe-se uma degradação em relação ao experimento anterior dado o maior volume de mensagens recebidas.

Já a manipulação de quatro veículos simultaneamente, como observado na Figura 5.5, faz com que haja muitos elementos na tela, sobrando pouco espaço para a interação com o mapa. Para esses casos, foi crucial utilizar o mecanismo de visualização em tela cheia para poder interagir com maior precisão. Por fim, o controle de quatro VANTs tornou mais evidente a queda de desempenho do sistema, tornando a experiência do usuário mais lenta tanto na parte de comando quanto na de visualização.

Figura 5.2 – Tela inicial da aplicação

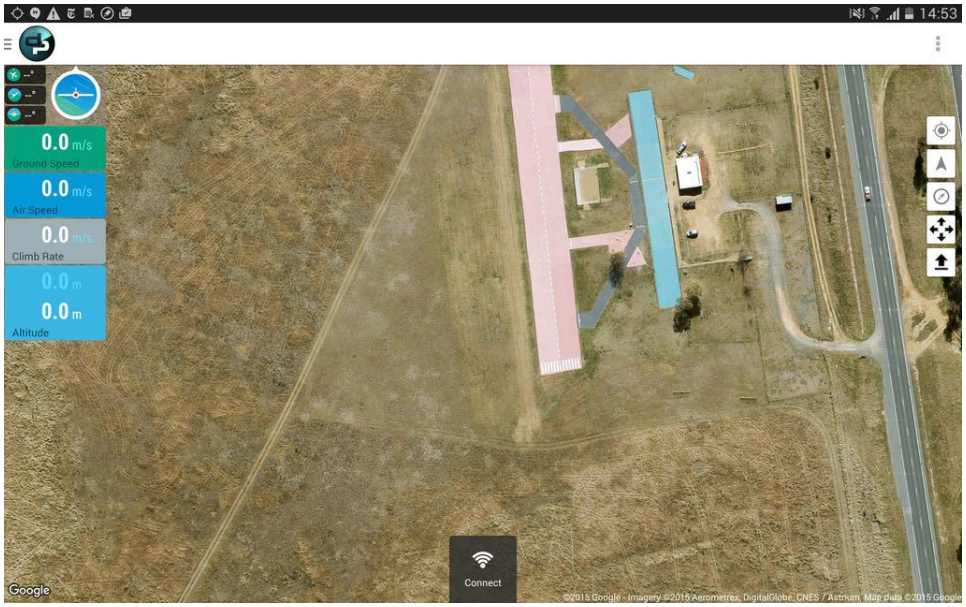


Figura 5.3 – Controle simultâneo de dois VANTs

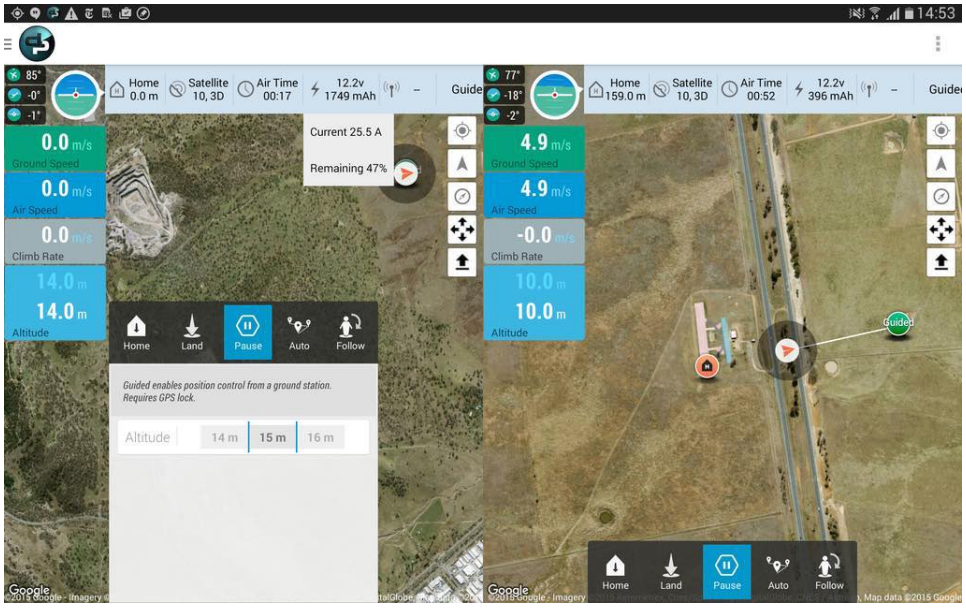
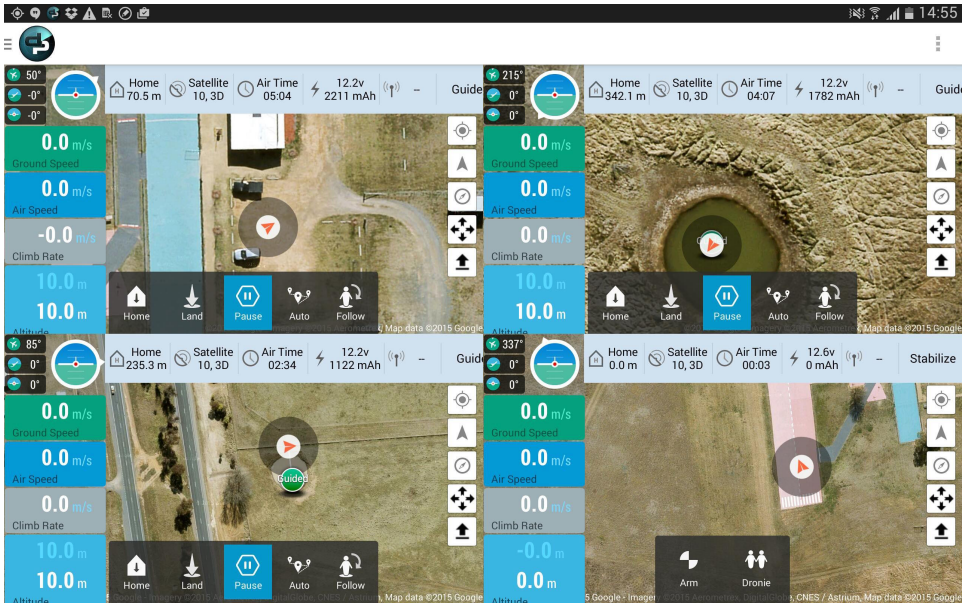


Figura 5.4 – Controle simultâneo de três VANTs

Figura 5.5 – Controle simultâneo de quatro VANTs



6 CONCLUSÃO

Este trabalho teve por objetivo o desenvolvimento de uma estação de controle que suportasse manipular múltiplos veículos aéreos não tripulados simultaneamente. Para isso, foi utilizado uma plataforma já existente e bastante conhecida para controle de apenas um VANT, chamada DroidPlaner, como ponto de início. A partir daí, foram então realizadas diversas modificações necessárias a fim de estender a plataforma para este projeto.

Inicialmente, foi preciso compreender todas as exigências que a nova aplicação requisitava em relação à comunicação. Para isso, optou-se pela utilização dos módulo *XBee*, visto que estes dispositivos permitem criar redes complexas e sofisticadas muito importantes para a troca de mensagens dos múltiplos VANTs.

Após compreender o funcionamento do DroidPlanner original e realizar as alterações internas, foram feitas as mudanças na interface com o usuário para suportar o controle de múltiplos veículos. As principais decisões foram feitas visando exibir todas as informações necessárias sobre os VANTs para o usuário, a fim de que este pudesse interagir com cada veículo individualmente, mas sem ignorar as demais informações dos outros veículos na tela. Para isso, optou-se pela divisão da interface original em diversos fragmentos.

Este projeto, na sua última versão, apresenta uma nova plataforma para controle e visualização de veículos aéreos não tripulados, contribuindo para suprir a necessidade de aplicativos deste gênero para o mercado.

6.1 Trabalhos futuros

Apesar de concluídas as transformações necessárias para este trabalho, ainda há muitas outras modificações e sugestões de problemas para dar continuidade ao seu desenvolvimento. Por isso, abaixo são citados alguns tópicos relevantes para trabalhos futuros.

6.1.1 Prevenção de colisões

A prevenção de colisões é um mecanismo crucial para a segurança dos VANTs quando mais de um veículo está sendo controlado. Esta aplicação, apesar de dar suporte à

visualização e ao controle, não oferece nenhum tipo de mecanismo que garanta que dois veículos não colidam, ficando a cargo do usuário esse cuidado. Por isso, estudos para tratar desse problema são relevantes nesse contexto, favorecendo um ambiente mais livre de acidentes e danos para todos.

6.1.2 Comunicação entre VANTs

Neste trabalho, toda a comunicação do VANT é feita única e exclusivamente com a plataforma de controle em *unicast*. A fim de realizar missões mais complexas ou até mesmo trocar informações relevantes, a implementação de um sistema de comunicação entre VANTs, sem a obrigatoriedade da comunicação com a estação de controle é um tópico bem interessante.

6.1.3 Múltiplos canais de Comunicação

Como detectado durante a experimentação, um único canal de comunicação para receber e enviar as mensagens acaba sobrecarregando a aplicação dado o alto volume de mensagens recebidas sequencialmente. Por isso, o estabelecimento de mais de um canal de comunicação com a plataforma de controle poderia ser uma solução possível para evitar esse gargalo, visto que a aplicação teria a possibilidade de tratar as mensagens recebidas em paralelo.

REFERÊNCIAS

ABDULLAH, Q. A. **Classification of the Unmanned Aerial Systems**, Geospatial Applications of Unmanned Aerial Systems (UAS), The Pennsylvania State University, 2014. Dispnível em: <https://www.e-education.psu.edu/geog892/node/5>

ALSHBATAT, A. L.; DONG, L., Performance Analysis of Mobile Ad Hoc Unmanned Aerial Vehicle Communication Networks with Directional Antennas. **International Journal of Aerospace Engineering**. Tafila, Jordânia, 2010

AMATO, A. **The 7 Best Agricultural Drones on the Market Today** 2014 Artigo disponível em: <http://dronelife.com/2014/10/01/best-agricultural-drones/>

AMAZON. **Amazon Prime Air**. Notícia disponível em: <http://www.amazon.com/b?node=8037720011>. Acesso em: nov. 2015.

ANDROPILOT. Aplicativo Andropilot disponível em <https://play.google.com/store/apps/details?id=com.geeksville.andropilot&hl=en>. Acesso em: nov. 2015.

APM. Documentação disponível em: <http://ardupilot.com/>. Acesso em: out. 2015.

ARDUPILOT. Código do ArduPilot disponível em: <https://github.com/diydrones/ardupilot>. Acesso em: jul. 2015.

BENEMANN, A.**Estação de Controle para Veículos Aéreos Não Tripulados**. Trabalho de Conclusão de Curso, Porto Alegre 2013

DEVICES. Lista de dispositivos compatíveis com o aplicativo DroidPlanner <https://github.com/DroidPlanner/Tower/wiki/Compatible-Devices>. Acesso em: abr. 2015.

DIGI. **XBee Gateway User's Guide**, 2014. Disponível em: <http://ftp1.digi.com/support/documentation/html/90001399/90001399_A/Files/XBee-concepts.html#_Toc384719514>. Acesso em: nov. 2015.

DIGI. **XBee / XBee-PROZigBee RF Modules User Guide**. 2015 <http://ftp1.digi.com/support/documentation/90000976_W.pdf> Acesso em: nov. 2015.

DOD. UAS Task Force, Airspace Integration Integrated Product Team **Unmanned Aircraft System Airspace Integration Plan** 2011. Disponível em: <http://www.acq.osd.mil/sts/docs/DoD_UAS_Airspace_Integ_Plan_v2_%28signed%29.pdf>

DPGIT. Código fonte do aplicativo Tower diponível em: <https://github.com/DroidPlanner/Tower> Acesso em: abr. 2015.

DPV1. Aplicativo DroidPlanner versão 1 disponível em: <https://play.google.com/store/apps/details?id=com.droidplanner&hl=en 2013>. Acesso em: abr. 2015.

DPV2. Aplicativo DroidPlanner versão 2 disponível em: <https://play.google.com/store/apps/details?id=org.droidplanner&hl=en>. Acesso em: abr. 2015.

DPV286. Código fonte do DroidPlanner versão 2.8.6 RC disponível em: <https://github.com/DroidPlanner/Tower/releases/tag/Droidplanner_v2.8.6_RC3>. Acesso em: abr. 2015.

GOOGLESTORE. Imagem disponível no site: <https://play.google.com/store/apps/details?id=org.droidplanner&hl=en> . Acesso em: nov. 2015.

IRIS Imagem disponível em: <http://store.3drobotics.com/products/iris>. Acesso em: nov. 2015.

MAVLINK. Documentação do protocolo MAVLink disponível em: <http://qgroundcontrol.org/mavlink/start> Acesso em: abril. 2015.

MICRODRONES. Aplicações de VANTs em resgates: <https://www.microdrones.com/en/applications/areas-of-application/search-and-rescue/> . Acesso em: nov. 2015.

MISSIONPLANNER. Aplicativo Mission Planner disponível em: <http://planner.ardupilot.com/>. Acesso em: nov. 2015.

PEPITONE, J. **Domino's tests drone pizza delivery**, 2013 Disponível em: <http://money.cnn.com/2013/06/04/technology/innovation/dominos-pizza-drone/>

PHANTOM. VANT para fotografia. Produto disponível em: <http://www.dji.com/product/phantom-3-pro>

PIXHAWK. Documentação disponível em: <https://pixhawk.org/modules/pixhawk>. Acesso em: out. 2015.

SAMSUNG. Imagem disponível em: <http://www.samsung.com/global/microsite/2014galaxynote10.1/image.html#>. Acesso em: nov. 2015.

SITL. Imagem disponível em: <http://dev.ardupilot.com/wiki/sitl-simulator-software-in-the-loop/>. Acesso em: nov. 2015.

SHELLBORN, A. **Drones are ready for take-off in the mining industry** Artigo disponível em: <http://www.miningandexploration.ca/technology/article/drones_are_ready_for_takeoff_in_the_mining_industry/> Fevereiro, 2015.

TOWER. Aplicativo Tower disponível em: <https://play.google.com/store/apps/details?id=org.droidplanner&hl=en>. Acesso em: abr. 2015.

XBEE. Imagem dispoível em: <http://examples.digi.com/get-started/basic-xbee-zb-zigbee-chat/>. Acesso em: nov. 2015.

XCTU. *Software* X-CTU disponível em: <http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu.>. Acesso em: out. 2015.