FUNDAÇÃO UNIVERSIDADE DO TOCANTINS
SISTEMAS DE INFORMAÇÃO

CÉLIA DA SILVA MORAIS

DESENVOLVIMENTO DE SISTEMA DE CONTROLE PARA VEÍCULO AÉREO NÃO TRIPULADO (VANT) POR MEIO DE DISPOSITIVOS MÓVEIS

PALMAS / TO
2014

CÉLIA DA SILVA MORAIS

DESENVOLVIMENTO DE SISTEMA DE CONTROLE PARA VEÍCULO AÉREO NÃO TRIPULADO (VANT) POR MEIO DE DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da UNITINS - Fundação Universidade do Tocantins, como requisito à obtenção do título de Bacharel em Sistemas de Informação.

Orientador Prof. Me. Alex Coelho

PALMAS / TO
2014

FICHA CATALOGRÁFICA
Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca da Universidade do Tocantins
Campus I

M827d Morais, Célia da Silva
Desenvolvimento de Sistema de Controle para Veículo Aéreo não Tripulado (VANT) por meio de Dispositivos Móveis. / Célia da Silva Morais – Palmas, 2014.
90f.

Monografia (TCC) – Universidade do Tocantins, Curso de Sistemas de Informação, 2014.
Orientador : Prof. Alex Coelho.

1. Android. 2. Arduino. 3. Comunicação USB. 4. Sistema de Controle. 5. VANT. I. Título.

CDD 003.3

Bibliotecário: Paulo Roberto Moreira de Almeida
CRB-2 / 1118

UNITINS

Coordenação do Curso de
Sistemas de Informação

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA FUNDAÇÃO UNIVERSIDADE DO TOCANTINS - UNITINS

Aos **15** dias do mês de **Dezembro** de **2014**, reuniu-se na Fundação Universidade do Tocantins, às **16 horas**, sob a Coordenação do Professor **Silvano Malfatti**, a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Alex Coelho** (Orientador) e Professor **Igor Yepes**, para avaliação da defesa do trabalho intitulado **"Desenvolvimento de Sistema de Controle para Veículo Aéreo Não Tripulado (VANT) por meio de dispositivos móveis"** da acadêmica **Célia S. Morais** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso (TCC). Após exposição do trabalho realizado pelo acadêmica e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: 9,5.

Sendo, portanto, o Acadêmico: (X) Aprovado ( ) Reprovado

Assinam esta Ata:

Professor Orientador: Alex Coelho

Examinador: Igor Yepes

Examinador: Silvano Maneck Malfatti

Acadêmico: Célia S. Morais

Professor Silvano Malfatti
Coordenador do Curso de Sistemas de Informação

Silvano Maneck Malfatti
Coordenação de Sist. de Informação
Matrícula 001558
Port/Unitins/GRE nº 362/20

“É do senso comum capturar um método e experimentá-lo.
Se ele falhar, admita isso com franqueza e experimente outro.
Mas, acima de tudo, tente algo”
Franklin Delano Roosevelt (1882-1945).

Às pessoas que dão significado à minha vida:

Valdenor,

Samuel, Valdenor junior

e Geisa Alana.

AGRADECIMENTOS

Agradeço, em primeiro lugar, à minha família: À meu esposo, pela paciência , apoio, confiança e incentivo. Meus Filhos (Samuel, Valdenor Junior e Geisa Alana), pela compreensão de cada momento em que não dediquei-lhes total atenção por esse período.

Quero agradecer à todas as pessoas que de alguma forma contribuíram para o desenvolvimento desse trabalho, e particularmente:

Aos professores Igor Yepes, Marco Antonio e Silvano Malfatti, pelo apoio, direcionamento, materiais disponibilizados, incentivo, lições aprendidas, dicas e todo o processo que possibilitou o início e sequencia desse trabalho.

Ao meu orientador, Alex Coelho, pela paciência, pelas inúmeras correções dos trabalhos, discussões, orientações e observações que favoreceram o aprendizado instigando o interesse pela pesquisa .

Aos colegas do grupo de pesquisa, João Paulo e Bruno Morais pela troca de experiências, informações, incentivo e discussões que resultaram em conhecimento adquirido.

À professora Luciane Fraga, por incentivar minha entrada no projeto de pesquisa no inicio do curso, com dicas e lições que puderam ser aplicadas durante todo o trajeto acadêmico.

Ao professor Claudio Castro, pelas lições, e principalmente pela motivação que não me deixou desistir do curso ainda no primeiro período. Motivação que de alguma forma serviu como estímulo nas dificuldades que vieram depois.

Às professoras Silvéria Basniak e Stéphany Martins, pelas lições, discussões e trocas de informações que tanto contribuíram para escrita e desenvolvimento dos trabalhos acadêmicos.

Obrigada à todos.

RESUMO

O avanço tecnológico e a utilização dos conceitos de mobilidade que se empregam nos dispositivos móveis caminham em paralelo. Hoje é impossível pensar em um mundo sem a facilidade de acesso que celulares e *tablets* oferecem. A cada dia o acesso a internet e a transmissão de dados ficam mais fáceis e transparentes. A tecnologia permite avanços em diversas outras áreas importantes como a visão computacional e a inteligência artificial, o que abre um vasto leque a ser explorado, como no caso dos Veículos Aéreos não Tripulados (VANTS). Nesse contexto, esse trabalho propõe a implementação de um software para dispositivos móveis que possibilite o controle de um VANT do tipo quadrotor, com base na plataforma Arduino, por meio de comunicação sem fio. Para desenvolvimento do controle foram realizados estudos de trabalhos relacionados para compreensão da aerodinâmica dessas aeronaves. O ponto principal do trabalho se ateve a comunicação do dispositivo móvel com a plataforma de hardware Arduino por meio de protótipos, divididos entre simulações variadas de comunicação USB e sem fio. O software proposto é composto por dois controles *joysticks* que permitem o envio de até dezesseis comandos para a plataforma Arduino, o que atende à necessidade dos movimentos básicos da aeronave de maneira que possa controlá-la.

Palavras-Chave: Android, Arduino, Comunicação USB, Sistema de Controle, VANT.

ABSTRACT

The technological advancement and the using of mobility that are used in mobile devices walking in parallel. Today is impossible imagine a world without the easy access to phones and tablets offers. Each day the access of internet and the data transmission are more easing and transparent. the technology allow advancing in variety of other important areas such as computer vision and artificial intelligence, which opens a wide range to be explored, as in the case of unmanned aerial vehicles (UAV). In this context, this work proposes an implementation of a software for mobiles devices which enables the control an UAV quadrotor type, with base in the platform Arduino,through wireless communication. For control deployment were performed studies related works for understanding the aerodynamics of these aircraft. the principal point of work stuck to the mobile communication with the Arduino hardware platform through prototypes, simulations divided between various USB and wireless communication. The proposal software is composite by two joysticks controls that allow of up to sixteen commands to the Arduino platform, that meet the needs of basic movements of the airplane from mode that may control it.

Key-words: Android, Arduino, Control System, UAV, USB communication.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AC	Alternating Current
ACM	Abstract Control Model
ANAC	Agencia Nacional de Aviação Civil
API	Application Programming Interface
BVLOS	Operação Alem da Linha Visada Visual
CBT	Companhia brasileira de tratores
CDC	Communications Device Class
CPU	Central Processing Unit
CTA	Centro Tecnológico da Aeronáutica
DOF	Degree Of Freedom
EDR	enhanced data rate
EEPROM	Electrically Erasable Programmable Read-Only Memory
GHz	GigaHertz
GND	Ground
GPL	General Public License
GPS	global positioning system
HUD	Head Up Display
IAPP	Inter-Access-Point Protocol
IDC	International Data Corporation
IDE	Integrated Development Environment
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
JVM	Java Virtual Machine
KB	KiloByte
KHZ	quilo-hertz
LED	Light Emitting Diode
LGPL	Lesser General Public License
mA	miliampère
MAVLINK	Micro Air Vehicle Link
Mbps	Megbit por Segundos
MHZ	mega-hertz
mW	megawatt

PCB	PRINTED CIRCUIT BOARD
PWM	PULSE-WIDTH MODULATION
RF	RADIOFREQUENCY
RPA	AERONAVE REMOTAMENTE PILOTADA
RPAS	SISTEMA DE PILOTAGEM DO RPA
RPS	ESTAÇÃO DE PILOTAGEM REMOTA
RPV	REMOTE PILOTED VEHICLE
SDK	SOFTWARE DEVELOPMENT KIT
SRAM	STATIC RANDOM ACCESS MEMORY
TCP/IP	TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL
UDP	USER DATAGRAM PROTOCOL
USB	UNIVERSAL SERIAL BUS
USB-OTG	USB ON-THE-GO
WEP	WIRED EQUIVALENT PRIVACY
WPA	WI-FI PROTECTED ACCESS
WPA2	WI-FI PROTECTED ACCESS II
WPAN	WIRELESS PERSONAL AREA NETWORKS
VANT	VEÍCULO AÉREO NÃO TRIPULADO
VLOS	OPERAÇÃO EM LINHA VISADA VISUAL
XML	EXTENSIBLE MARKUP LANGUAGE

SUMÁRIO

1. INTRODUÇÃO

A evolução da tecnologia móvel é responsável pelo avanço em diferentes aspectos da mobilidade. De acordo com pesquisas apresentadas pela BRASSCOM (2014), a maioria das pessoas passam no mínimo 149 minutos por dia ligadas aos seus aparelhos móveis, seja por motivos profissionais ou necessidade de comunicação e entretenimento. Essa prática ocorre devido a possibilidade de comunicação e manipulação de recursos computacionais a qualquer momento, em qualquer lugar e em situações diversas, o que resulta em mudança de comportamento e necessidades relativamente diferentes entre as pessoas (PROMON, 2014).

Toda essa evolução da tecnologia móvel é possível, dentre outros fatores, graças ao avanço no universo de aplicações móveis direcionadas às interações em tempo real. Diante desse fato, a ideia de "Estação de Trabalho", ou mesmo de "*Desktop*" passa a ser irrelevante para o mundo atual, visto a possibilidade da realização de muitas atividades por meio de dispositivos móveis. A mobilidade hoje é uma necessidade primordial no cotidiano, em que se espera acesso imediato às novas soluções apresentadas com o fim de possibilitar tomadas de decisões com a máxima flexibilidade e em tempo hábil (BRASSCOM, 2014).

Neste contexto de possibilidades diversas de aplicações para dispositivos móveis, sua utilização como dispositivo de controle já é observada no contexto residencial, como apresentado na Figura 1, conhecida como domótica[1], e mesmo comercial, com destaque para soluções de mobilidade.

Figura 1-1 - Controle para casas inteligentes.
Fonte: AUSTRALTECH, 2014

[1] Tecnologia que permite gestão dos recursos habitacionais.

Essa concepção especial dos dispositivos móveis, aliados a evolução em campos importantes da tecnologia como a visão computacional e da própria inteligência artificial, integram uma dimensão versátil em pleno crescimento a serem exploradas, como no caso dos VANTs (Veículos Aéreos Não Tripulados).

Os VANTs, também conhecidos como Drones, referem-se a aeronaves projetadas para missão de voos sem a presença de um piloto a bordo. Apesar de parecer um aeromodelo de brinquedo para algumas pessoas, essas aeronaves trazem diversas tecnologias envolvidas, como comunicação e localização georreferencial, o que possibilita elaborar soluções inovadoras. A utilização de VANTs se populariza rapidamente no uso civil, frente ao cenário comercial em que a aquisição ou mesmo desenvolvimento e construção tornam-se viáveis economicamente (FURTADO, 2008).

O uso de dispositivos móveis como sistema de controle remoto, torna-se possível junto ao desenvolvimento de soluções para VANTs, pois permitem, entre tantos recursos, o tráfego de dados por rede sem fio através de um sistema já acoplado em um dispositivo no próprio VANT.

A atividade essencial do desenvolvimento de um sistema de controle se baseia na problemática do mapeamento entre as variáveis que definem o comportamento do VANT frente a sua interface de manipulação, definida por meio de software. Isso passa a ter maior impacto considerando as peculiaridades existentes em um ambiente mais restrito como o dos dispositivos móveis, com ferramentas e recursos limitados.

Depois de analisar estes princípios é essencial a definição do modelo em que ambos os dispositivos, no caso o VANT e *Smartphone* ou *tablets*, tenham assegurado a função de espaço de estados que caracterizem os movimentos em um limite geográfico. Isso com os mecanismos de controle com a eficácia necessária para manter a estabilidade e direcionamento para a execução das tarefas (FERREIRA, 1998).

Frente a essas considerações sobre os VANTs, aliado às possibilidades que são pertinentes aos dispositivos móveis, verifica-se que a integração de plataformas e instrumentos tecnológicos, podem concretizar possibilidades diferenciadas para que sejam enfrentadas e trabalhadas nas mais diversas áreas

do conhecimento, com a utilização de sistemas embarcados em VANTs controlados por *tablets* ou *Smartphone's,* por meio de redes sem fio.

Com base nesses princípios, o trabalho objetiva o desenvolvimento de um software para dispositivos móveis que possibilite o controle de um VANT do tipo quadrotor com base na plataforma Arduino, por meio de redes sem fio. Os objetivos específicos são apresentados a seguir:

- Estudar e compreender os conceitos aerodinâmicos associados ao VANT do tipo quadrotor;
- Realizar o levantamento das ferramentas pertinentes ao desenvolvimento para dispositivos móveis;
- Buscar a configuração mais adequada em termos de custo/benefício;
- Desenvolver o sistema de controle que receberá os comandos do operador humano para um dispositivo móvel;
- Efetuar os testes com o software desenvolvido.

O trabalho está estruturado da seguinte forma:

- Capítulo 1 – Introdução;
- Capítulo 2 - Revisão de Literatura;
- Capítulo 3 - Metodologia;
- Capítulo 4 – Desenvolvimento do Sistema;
- Capítulo 5 - Conclusões.
- Capítulo 6 - Referências

A Revisão de Literatura está dividida em três partes: A primeira parte está relacionada aos conceitos do VANT, particularidades do tipo quadrotor que são indispensáveis para compor o projeto, referencial histórico sobre essas aeronaves e a situação legal para uso no Brasil ; A segunda parte do Capítulo dois apresenta a plataforma Arduino, base para construção de protótipos com baixo custo e tecnologia adaptável à necessidade de sua utilização; A terceira parte, discorre sobre a tecnologia Android, conceitos básicos sobre a plataforma de desenvolvimento e um resumo da história da evolução do sistema operacional Android para qual o software foi desenvolvido.

O Capítulo três apresenta a metodologia aplicada para o desenvolvimento da pesquisa, dificuldades encontradas, materiais utilizados durante o processo de criação de protótipos e os métodos aplicados para o desenvolvimento do software.

O Capítulo quatro se refere aos resultados com a implementação do sistema que está dividido em quatro subseções: A primeira, "Hardware - criação do protótipo de Hardware", mostra o processo de criação de protótipos com base na plataforma Arduino que tem o objetivo de simular a comunicação com o VANT e como essa comunicação entre o Arduino e o mundo exterior pode ser realizada. A segunda subseção, é apresentado o desenvolvimento do software que permite a comunicação com o Arduino, e detalha a implementação da interface do aplicativo. Por fim, no capítulo ainda são considerados diversos aspectos relevantes para o desenvolvimento como os obtidos com o sistema de comunicação entre o android e o Arduino, as dificuldades encontradas nos experimentos e descreve qual melhor performance de transmissão de dados foi estabelecida.

O Capítulo 5 se refere à conclusão do trabalho e faz referencia a trabalhos futuros, seguido pelas referências.

2. REVISÃO DE LITERATURA

Esse capítulo aborda conceitos e pontos importantes referentes aos três pilares que são a base para o desenvolvimento do projeto: O VANT, objeto motivador para criação do software de controle, a plataforma Arduino, que é a base de construção do VANT a ser manipulado e a plataforma Android, ponto essencial para a implementação do software.

2.1 VEÍCULO AÉREO NÃO TRIPULADO – VANT

2.1.1 CONCEITO

O termo VANT abrange uma variedade de veículos aéreos que tenham movimentação controlada por sistema remoto ou capacidade de voos pré-programados antes de seu lançamento. A terminologia VANT não se refere somente à aeronave em si, mas a todos os equipamentos de suporte utilizados pelo sistema que compõem suas características, como os sensores, microcontroladores, software, estação de solo e hardware de comunicação (BEARD; McLAIN, 2012).

De acordo com a Portaria Normativa do Exercito Brasileiro nº 606, do Ministério da Defesa, de 11 de junho de 2004, o conceito de VANT se resume em um veículo de pequeno porte, construído a partir de materiais de difícil detecção, com piloto remoto e que possua asas fixas ou rotativas. Seu objetivo é sobrevoar um alvo ou área de interesse para fornecer informações por meio de seu sistema de vigilância eletrônica. O artigo 4 da mesma portaria especifica o VANT como uma plataforma aérea de baixo custo operacional que pode ser operada por controle remoto ou executar perfis de voo de forma autônoma. Suas funções variam entre várias possibilidades como o transporte de cargas, utilização de sensores diversos e equipamentos de comunicação ou mesmo servir como alvo aéreo e conduzir designador de alvo e cargas letais, com fins bélicos.

2.1.2 HISTÓRIA DOS VANTS

Apesar da popularização rápida e recente, os VANTs não se enquadram como novidade tecnológica. O conceito VANT foi utilizado pela primeira vez em

1849, na ocorrência de um ataque do Exercito Austríaco à cidade de Veneza, na Itália. Tratava-se de balões carregados de munições com alto poder de destruição com finalidade de sobrevoar a cidade e programados para explodir sua carga após determinado tempo do ponto de partida. A maioria dos balões atingiam o alvo, porém alguns voltavam para a base, devido as mudanças temporais repentinas (HARDGRAVE, 2014).

Essa mesma ideia dos balões munidos de bombas e programados para explodir quando atingissem o alvo, foi utilizada pelos Exércitos do Sul e do Norte durante a Guerra Civil Americana de 1861 e na primeira Guerra Mundial, em 1914. Em 1916, os norte-americanos Lawrence e Sperry construíram um aeromodelo, que segundo (NONAMI et al., 2010) voou por cerca de 48 km e ganhou o nome de “torpedo-avião”.

Antes, em 1914, Lawrence já tinha apresentado a invenção em uma Competição de Segurança de Avião, com um aparelho estabilizador, acoplando um giroscópio para controlar a superfície e manter o eixo de voo da aeronave, que possuía piloto automático. A demonstração de estabilização com o giroscópio se deu quando Lawrence e seu companheiro se penduraram nas asas para causar mudança de altitude e desestabilização (NONAMI et al., 2010).

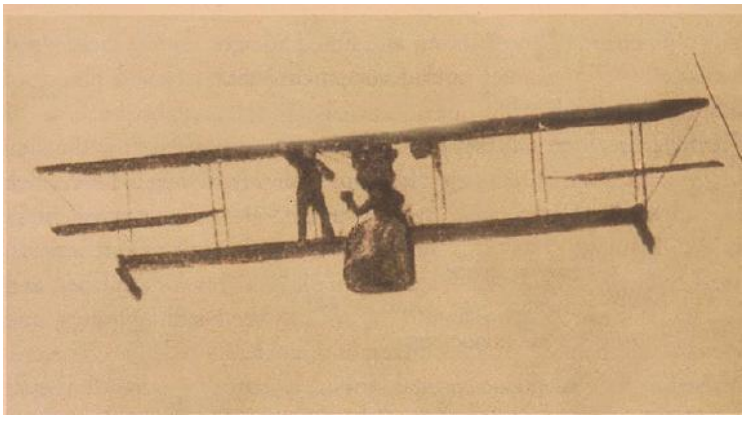


Figura 2-1- Demonstração de estabilização de voo em 1914.
Fonte: AMERICAN GENESIS, 2014.

O emprego do giroscópio nas aeronaves constituiu um avanço significativo na construção de VANTs. Em 1935, o norte-americano Reginald Denny projetou e testou o RP-1 ou RPV (*Remote Piloted Vehicle*), controlado por rádio. Foi então que se iniciou uma busca pelo aperfeiçoamento de maneira que nos anos seguintes nasceram o RP-2 e RP-3. O mais completo RPV, o RP-4, foi concluído

em 1939. O Exercito dos Estados Unidos da América requisitou 53 unidades desses veículos e designou-os como OQ-1 (HARDGRAVE, 2014).

No Brasil, o primeiro VANT registrado foi o BQM-1BR em 1982. Essa aeronave podia atingir uma velocidade de 530 km/h e chegava ao limite de 6500 metros de altura. Sua função era ser alvo aéreo para treinamentos diversos em substituição ao similar norte-americano, então em uso na época que possuía custo muito elevado (PAULA, 2014). A Figura 2-2 mostra o protótipo do BQM-1BR.

Figura 2-2 - BQM - 1BR - Protótipo do Primeiro VANT Brasileiro.
Fonte: PAULA, 2014.

O BQM-1BR era um veículo à jato, remotamente tripulado, totalmente brasileiro, pois foi planejado e construído no país. O projeto foi arcado pela CBT (Companhia Brasileira de Tratores), com seus próprios recursos somado aos gastos com o desenvolvimento da turbina "Tietê TJ-2", como era conhecida, que foram de responsabilidade da CTA (Centro Tecnológico da Aeronáutica) (PAULA, 2014).

2.1.3 USO DE VANTS NO BRASIL

As aplicações para VANTS de pequeno porte são inúmeras. Entre as mais comuns encontram-se a navegação *indoor*, mapeamentos, acesso a ambientes hostis e perigosos, supervisionamento, obtenção de imagens, agricultura, segurança pública, serviços de entrega e construção civil (BEARD; McLAIN, 2012).

No Brasil, o uso de VANT's é restringido por burocracias, porém, a Polícia Federal Brasileira está entre os principais usuários dessa tecnologia em todo o

mundo. O Gerente do projeto VANT da Polícia Federal, agente Álvaro Marques afirma que "em termos de segurança pública, o Brasil está um pouco a frente de outros países" (OTOBONNI, 2014). O aparelho utilizado pela Polícia Federal é o *Heron*, de produção israelense e suas dimensões chegam quase a mesma de um planador. A Figura 2-3 mostra um *Heron* em pleno voo (OTOBONNI, 2014).

Figura 2-3 - VANT Heron.
Fonte: FENAPEF, 2014

Embora o uso das aeronaves por civis tenham aumentado rapidamente e o VANT esteja cada vez mais popular, no Brasil ainda não existe uma legislação específica para sua utilização. Em fevereiro de 2014, a ANAC- Agencia Nacional de Aviação Civil apresentou uma proposta de regulamentação para VANTS. Um resumo dos níveis de regulamentação proposto pode ser visto no Quadro 1.

Quadro 1 - Níveis de regulamentação proposto pela ANAC.

	Indoor		Área privada aberta		Área pública aberta		Áreas desabitadas	
	Área Privada	Área Pública	Até 400ft VLOS	> 400ft, BVLOS	Até 400ft VLOS	> 400ft, BVLOS	Até 400ft VLOS	> 400ft, BVLOS
Aeromodelo	Básica	X	Básica	X	1	X	Básica	X
RPA 25kg	Básica	1	Básica	1	1	X	Básica	1
RPA 25-150kg	X	X	2	2	X	X	2	2
RPA > 150kg	X	X	3	3	X	X	3	3
VANT autônomo	X	X	X	X	X	X	X	X

Fonte: ANAC, 2014.

Os níveis de exigências para operações estão classificados como:

- Básica: Nenhuma exigência para operar
- 1 - Nível de exigência baixo;
- 2- Nível de exigência intermediário;
- 3- Nível de exigência alta;
- X- Operação proibida.

A título de esclarecimento, a ANAC definiu alguns pontos para a proposta de regulamentação conforme Quadro 2.

Quadro 2 - Definições conforme ANAC.

QUADRO II		
SIGLA	**Nome**	**Definição**
Aeromodelo		Toda aeronave não tripulada com finalidade de esporte, lazer ou competição
VANT	Veículo Aéreo Não Tripulado	Toda aeronave não tripulada com finalidade diversa de esporte, lazer e competição.
RPA	Aeronave Remotamente Pilotada	VANT destinado à operação remotamente controlada.
RPS	Estação de Pilotagem Remota	Estação na qual o piloto remoto exerce suas funções e onde estão instalados todos os equipamentos e instrumentos de voo.
RPAS	Sistema de Pilotagem de RPA	Todo o conjunto de elementos abrangendo uma RPA, a EPS correspondente, os enlaces de comando e controle requeridos e quaisquer outros elementos que podem ser necessários a qualquer momento durante a operação.
Operação Remotamente Controlada		Operação na qual é possível a intervenção do piloto remoto em qualquer fase do voo
Operação Autônoma		Operação na qual não é possível a intervenção do piloto remoto no voo ou parte dele.
Piloto Remoto		Pessoa que manipula os controles de voo de um VANT ou aeromodelo.
VLOS	Operação em Linha de Visada Visual	Operação diurna na qual o piloto remoto mantém constante contato visual direto com o VANT ou o aeromodelo.
BVLOS	Operação Além da Linha de Visada Visual	Operação que não atenda às condições VLOS.

Nota: Os dados do Quadro estão dispostos na proposta de regulamentação para VANTs apresentada pela ANAC em fevereiro/2014.

Segundo a ANAC, o normativo proposto ainda será submetido à audiência pública, e que qualquer pessoa interessada poderá encaminhar contribuições. Essas contribuições serão analisadas e só depois serão liberadas para Diretoria Colegiada da ANAC.

2.1.4 O QUADROTOR

Existem vários modelos de VANTs, cada um com suas particularidades. O grande avanço tecnológico nessa área se aplica às aeronaves multi rotores, que permitem asas rotativas. Diferente dos veículos apresentados até aqui, os VANTs multi rotores tem levitação garantida por motores independentes (OLIVEIRA, 2005). A Figura 2-4 mostra uma aeronave multi rotor de 4 asas rotativas.

Figura 2-4 - VANT do tipo Quadrotor.
Fonte: BENEMANN, 2013.

Dentre os modelos multi rotores, o quadrotor se destaca cada vez mais em pesquisas e desenvolvimento.Trata-se de uma aeronave dotada de quatro asas rotativas e seus motores são posicionados de forma que o empuxo gire na mesma posição, o que permite um voo vertical, estagnado e de baixa velocidade com poder de manobras diversas (SÁ, 2012).

Segundo (LEISHMAN, 2006), o primeiro quadrotor não era VANT, pois tinha tamanho suficiente para carregar o piloto, porém existia dificuldade em estabilizar a aeronave o que resultava em voos mal sucedidos. Esse contratempo fez com que a ideia de trabalhar com quadrotor fosse abandonada. Só na década de 80 é que o quadrotor foi reinventado.

O VANT tem capacidade de autocontrole através da implementação do sistema com dispositivos de radiofrequência. Porém, para uma abordagem diferente, é interessante a utilização de um controle remoto, tanto para controlar melhor o veículo, como para aumentar seu desempenho (VASCONCELOS, 2013).

2.1.5 O SISTEMA

Os VANTs possuem propulsão própria através de forças aerodinâmicas que provocam a sua sustentação e não possuem cabine de pilotagem, utilizando para isso visão computacional ou outros meios que possibilitem ser controlada a distância. Geralmente isso se dá por meio de redes sem fio, com a intervenção humana através de *gadget* ou ainda com algoritmos de voo sem a intervenção humana (SOUSA, 2011).

De um modo geral, o sistema de piloto automático possui um computador central que designa seus movimentos. O sensor de posição ou movimento precisa a posição atual da aeronave e de acordo com os dados informados pelo piloto ou por um computador de navegação, realiza-se a correção necessária através dos servo motores e atuadores.

Oliveira (2005), explica que apesar das variações existentes entre os sistemas de VANTS, a maioria são compostos por três subsistemas: O subsistema do VANT- que é o próprio VANT com suas plataformas de diversos tamanhos, características e possibilidades de empregabilidade, o subsistema de Comando e Controle, e o subsistema de Lançamento e Recuperação.

O subsistema de Comando e Controle é responsável pelo controle do VANT, condução do lançamento e recuperação, e pela interpretação dos dados coletados pelos equipamentos a bordo. Já o subsistema de Lançamento e Recuperação é responsável pela decolagem e recuperação em segurança da aeronave (OLIVEIRA, 2005).

2.1.5.1 FUNCIONAMENTO E SISTEMA DE CONTROLE

Os sistemas de navegação e voo de um VANT, em sua maioria, se baseiam em controles de estado não linear, com 6 graus de liberdade, definidos

pelo DOF (*Degree of Freedom*). Isso, conforme a complexidade assumida para o voo, faz com que diversos níveis de modelagem sejam necessários, garantindo um maior ou menor grau de liberdade quanto aos deslocamentos e rotações que serão obtidas nos eixos x, y e z (OLIVEIRA, 2012), como pode ser observado na Figura 2-5.

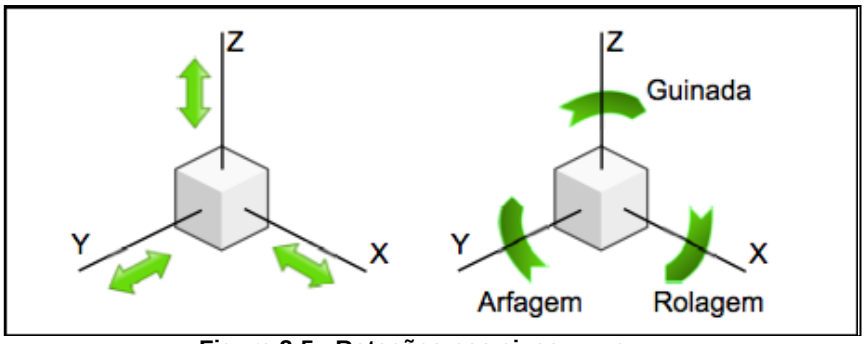


Figura 2-5 - Rotações nos eixos x, y e z.
Fonte: OLIVEIRA, 2012.

Conforme apresentado, o quadrotor é constituído de quatro motores e dois pares de hélices usados para girarem no sentido inverso um do outro, no qual os motores permitem o funcionamento da aeronave e voo. De acordo com (LEISHMAN, 2006) a disposição correta dos motores elimina o efeito dos *torques*, mas para funcionar, as hélices adjacentes giram em sentidos opostos.

A implementação de um sistema de controle para um quadrotor pode ser menos complexa em relação à outros modelos de VANTs. Segundo (GUIMARÃES, 2012) o sistema para um quadrotor possui 4 variáveis de entrada, que se referem aos quatro motores, e seis variáveis de saída, que são x, y, z, roll, pitch e yaw, que estão relacionadas aos ângulos de rotação, conforme Quadro 3.

Quadro 3 - Variáveis de movimentos dos VANTS.

QUADRO III		
VARIÁVEIS	**DESCRIÇÃO TÉCNICA**	**DESCRIÇÃO POPULAR**
Roll	Nome dado ao movimento em torno do eixo horizontal, na direção do eixo longitudinal.	Rolagem
Pitch	Nome dado ao movimento em torno do eixo horizontal, perpendicular ao eixo longitudinal	"Levantar o nariz" ou arfagem
yaw	Nome dado ao movimento em torno do eixo vertical,perpendicular ao eixo longitudinal	Guinada

Nota: Os dados para construção da Quadro foram baseados no estudo de Guimarães(2012).

Exemplo para movimentar a aeronave para frente, seria a necessidade dos motores 2 e 4 manterem a força, e o motor 3 aumentar a força em relação ao motor 1, conforme pode ser visto na Figura 2-6(a). Se fizer ao contrário com

relação aos motores 1 e 3, o quadrotor se movimenta pra traz, Figura 2-6(b). Se o movimento necessário for para direita ou esquerda, os motores 1 e 3 devem manter a mesma velocidade, e a alteração acontece nos motores 2 e 4 respectivamente, Figura 2-6(c e d) (SÁ, 2012).

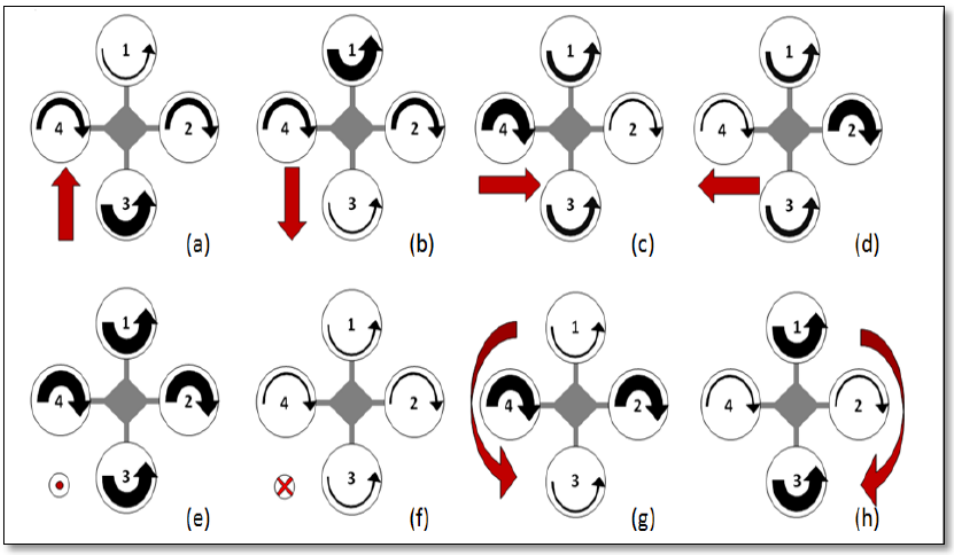


Figura 2-6 - Rotações de um quadrotor.
Fonte: Sá, 2012.

Na Figura 2-6 observa-se que as setas pretas indicam a rotação das hélices e as setas vermelhas representam o resultado do movimento. Além dos quatro movimentos básicos, a aeronave ainda é capaz de se movimentar verticalmente. Para isso, pode aumentar ou diminuir igualmente a velocidade de todos os motores ao mesmo tempo como exposto na Figura 2-6(e, f). Para girar no sentido horário, os motores 1 e 3 aumentam sua força enquanto os motores 2 e 4 diminuem. Se fizer ao contrário, o quadrotor gira no sentido anti-horário (SÁ, 2012).

Pensar no projeto que controle todas essas variáveis aliadas ao acoplamento de variáveis e as linearidades referentes a aerodinâmica, acionamentos e muitas outras diversidades de detalhes, é uma tarefa complicada.

De acordo com (BENEMANN, 2013), toda a problemática com as variáveis diversas para o controle e navegação de um VANT pode ser solucionado com a agregação de bibliotecas de implementação. Dentre elas há a MAVLink que pode

ser agregada a projetos. Trata-se de um protocolo de comunicação criado especialmente para aeronaves não tripuladas, ou tripuladas remotamente.

O protocolo de comunicação MAVLink está presente em diversos trabalhos relacionados à projetos com VANTs como: ARDUPILOT, PX4FMU, SMARTAP, MATRIXPILOT e DROIDPLANNER. Está liberado para uso sobre a licença LGPL(*Free Software Fundation*) desde 2009 e sua principal característica é a transferência de dados do VANT, como posição e orientação. Toda comunicação é realizada por meio de pacotes pré definidos em uma arquivo XML (BENEMANN, 2013).

2.2 ARDUINO

Consiste em uma plataforma aberta para prototipação eletrônica. Trata-se de uma plataforma flexível e fácil de usar, formada por dois componentes principais: hardware e software (LEMOS, 2014).

Figura 2-7 - Arduino - Placa Microcontroladora e Ambiente de Desenvolvimento.
Fonte: Elaborada pela autora.

O hardware Arduino é formado por uma placa de prototipagem utilizada para os projetos e o software se refere à uma IDE(*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado) utilizada para programar os códigos que serão embarcados no hardware (FBS, 2014).

2.2.1 ARDUINO - O HARDWARE

A placa Arduino trata-se de um circuito microcontrolador. Possui 6 entradas analógicas em que cada uma se liga a um conversor analógico-digital de 10 bits, o que permite que a leitura analógica se transforme em um valor entre 0 e 1024 . Sua voltagem varia de 0 a 5V, de modo que pode ser alterado conforme a necessidade da aplicação (BRESSAN, 2014). As principais características da placa Arduino são apresentadas no Quadro 4.

Quadro 4 - Principais Características da placa Arduino.

QUADRO IV	
NOME	**CARACTERÍSTICA**
Microcontrolador	ATmega328 ou ATmega168
Tensão de alimentação	7 - 12 V
Formas de alimentação	Porta USB ou adaptador AC com pino redondo de 2,1 mm e centro positivo(9V).
Pinos I/O digitais	14 - de 0 a 13
Pinos de entrada analógica	6 - de 0 a 5
Pinos de saída analógica	6 (3, 5, 6, 9, 10 e 11) - modulação PWM
Corrente contínua do pino I/O	40 mA
Corrente contínua para o pino 3.3 V	50 mA
Memória flash	32 KB
SRAM	2 KB
EEPROM	1KB
Frequência de *clock*	16 MHz

Fonte: Adaptação de (BRESSAN, 2014).

A Figura 2-8 mostra a placa Arduino com indicação de seus componentes.

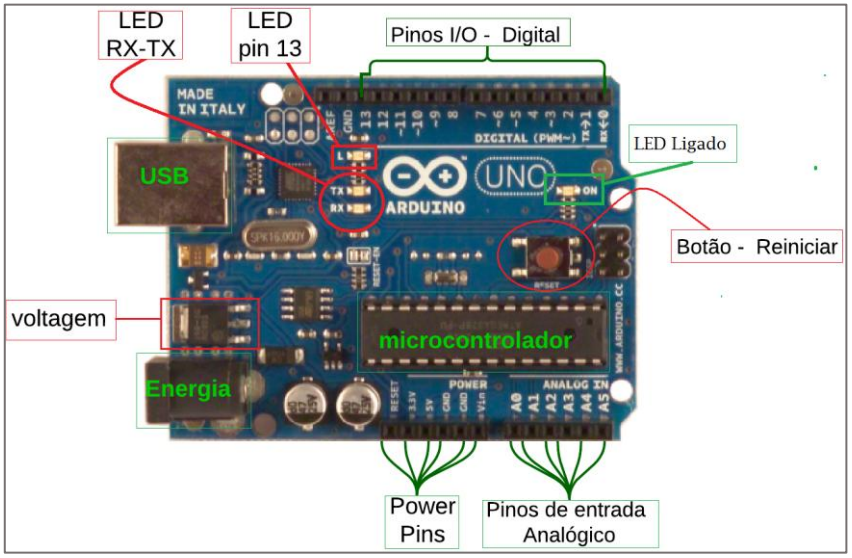


Figura 2-8 - Arduino com Descrição dos Componentes.
Fonte: Elaborada pela autora.

A plataforma Arduino possui componentes complementares que propiciam a programação e possibilitam a incorporação para outros circuitos. Os conectores são expostos de maneira que permitem a interligação de módulos expansivos à CPU da placa Arduino. Projetos e esquemas de hardware da plataforma são distribuídos sob a licença *Attribution Creative Commons Share-Alike* 2.5.

2.2.2 O SOFTWARE - IDE ARDUINO

O software Arduino se refere ao Ambiente de desenvolvimento (IDE) que permite a criação de *sketchs* da qual será feita *upload* para a placa Arduino. O IDE Arduino é escrito em Java e inclui o editor de código capaz de corrigir, compilar e carregar programas para a placa Arduino somente com um clique.(FBS, 2014). O código a ser embarcado na placa Arduino tem a extensão *.ino e possui uma estrutura básica dividida em dois blocos de funções indispensáveis para o funcionamento correto do hardware Arduino:

- setup() - Configuração inicial. É aqui que são inicializados os pinos I/O e a comunicação serial.
- loop() - Execução das tarefas.

A Figura 2-9 mostra o ciclo básico para programação Arduino.

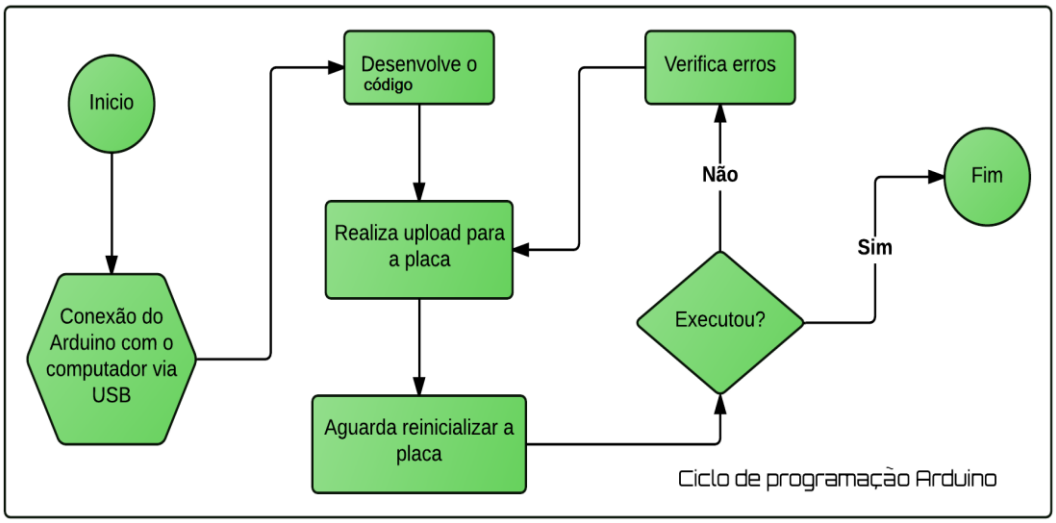


Figura 2-9 - Ciclo básico para programação Arduino.
Fonte: Elaborada pela Autora.

O código fonte para IDE e bibliotecas de funções da placa estão protegidos sob a licença GPLv2, e hospedadas pelo projeto *Google Code*. Essa licença garante a liberdade dos usuários para usar, modificar e distribuir as modificações para qualquer fim, mesmo que seja para fins comerciais.

2.2.3 COMUNICAÇÃO

Em sua maioria a comunicação entre a plataforma Arduino e o mundo exterior se dá por meio da porta serial. Assim é possível realizar a comunicação entre a placa microcontroladora e um programa externo ao ambiente de desenvolvimento Arduino, por exemplo, somente com um cabo USB, o mesmo que utilizado para ligar o aparelho ao computador, que traduz os bits para a porta serial. Essa comunicação pode ser feita entre qualquer microcontrolador habilitado para conversas nos parâmetros da comunicação serial (SILVA, 2013).

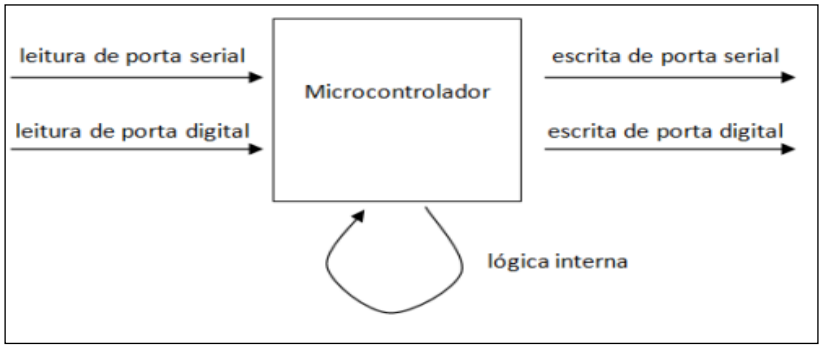


Figura 2-10 - Exemplo de Comunicação pela Porta Serial - Arduino.
Fonte: SILVA, 2013.

A Arduino UNO tem duas portas separadas para comunicação serial, que são respectivamente as portas 0 e 1, ou seja, a RX(*receive*) e TX(*transfer*). Toda comunicação serial acontece por essas portas (SILVA, 2013).

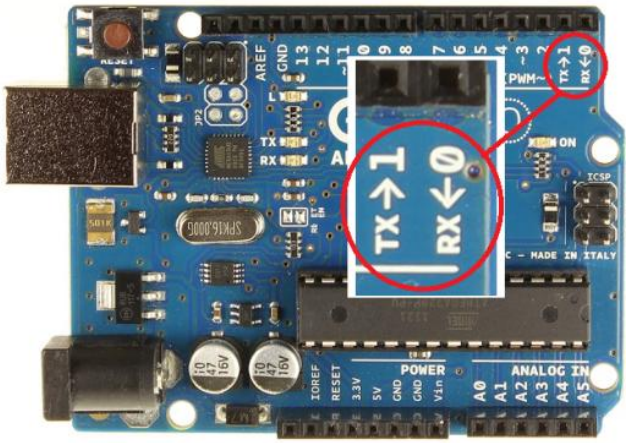


Figura 2-11 - Portas de Comunicação da Arduino.

A comunicação por meio de uma rede sem fio refere-se à transmissão de mensagens pelo meio sem a necessidade de qualquer tipo de cabos. Para que essa comunicação funcione com a Arduino é utilizado módulos para Bluetooth, WI-FI e radiofrequência, em que os bits são interpretados para a porta serial, de maneira transparente para o usuário. Cada meio de comunicação tem suas particularidades e um módulo adequado para ser acoplado à placa Arduino. Apesar da comunicação acontecer sem a necessidade de fios, a interpretação pela porta serial utiliza os parâmetros da conexão USB.

2.2.3.1 BLUETOOTH - PADRÃO IEEE 802.15

O Bluetooth é um protocolo para redes sem fio de curto alcance (*Wireless personal area networks* - WPAN). O padrão IEEE 802.15 é projetado para baixo consumo de energia, com base em *microchips* transmissores de custo reduzido em cada dispositivo e é bastante simples e eficiente para acionar motores, relés, e outros dispositivos em curta distância (TANENBAUM, 2011).

Para fazer a comunicação entre um dispositivo móvel e a plataforma Arduino é necessário conectar um comunicador com módulo Bluetooth de 5V na placa Arduino e fazer a conexão pela porta serial. No caso dos VANTs em particular, o Bluetooth tem pouca aplicação, já que a aeronave precisa de um alcance maior para voar (BENEMANN, 2013).

O módulo produzido pela empresa Microchip integra toda funcionalidade indispensável para a utilização via Bluetooth. A Figura 2-12 mostra o módulo RN-42.

Figura 2-12 - Módulo para Comunicação via Bluetooth.
Fonte: MICROCHIP, 2013.

O Módulo RN-42 suporta múltiplos protocolos de interface e possui configuração simples. Além de ser um produto certificado, é compatível com produtos de séries anteriores, o que faz dele uma solução completa para Bluetooth. Possui suporte para Bluetooth EDR PCB de alta Performance e oferece até 3 Mbps de taxa de transmissão de dados para distancias de até 20 metros (MICROCHIP, 2013).

2.2.3.2 WI-FI - PADRÃO IEEE 802.11

WI-FI é uma rede sem fio baseada no padrão IEEE 802.11. É a conexão mais popular atualmente e trabalha numa frequência de 2,4 ou 5,8 GHz, com potência que varia de 1mW a 1W. A rede WI-FI tem taxa de comunicação entre 1Mbps e 600Mbps e conta com disponibilidade de mecanismos de segurança como WEP,WPA, WPA2 (TANENBAUM, 2011). O Quadro 5 mostra as principais classificações do padrão IEEE 802.11.

Quadro 5 - Características do padrão IEEE 802.11.

QUADRO V				
Padrão IEEE	**Velocidade padrão (Mbps)**	**compatibilidade**	**Frequência (GHz)**	**MAX utilizadores**
802.11a	54	11b e 11g	5	64
802.11b	11		2,4	32
802.11g	54	11b	2,4	32
802.11d	Permite o padrão 802.11 operar em países onde não tem compatibilidade.			
802.11e	Agrega qualidade de serviço(QoS) às redes 802.11			
802.11f	Define o protocolo IAPP(Inter-Access-Point Protocol)			
802.11h	Versão do padrão 11a compatível com a banda 5GHZ da Europa			
802.11i	Traz as primitivas de segurança aos protocolos 802.11b, 802.11a e 802.11g.			
802.11j	Opera nas faixas 4.9 GHz e 5GHz, no Japão			
802.11k	Padrão para indústrias que permite transmitir dados de gerenciamento			
802.11r	Padroniza o *hand-off* (entrega/transição) rápido			
802.11s	Padroniza "*self=healing/self-configuring*" nas redes *mesh*			
802.11u	Interoperabilidade com outras redes móveis/celular			

Fonte: Elaborada pela Autora.
Nota: Os dados da tabela foram levantados com base no Livro "Redes de Computadores" de TANEMBAUM, 2011

A comunicação entre o dispositivo móvel e a plataforma Arduino através de uma rede Wi-Fi pode ser feita de diversas formas. Dentre os padrões mais comuns está a porta serial "paralela", que pode ser utilizado em comunicação serial com a RS-232 para Ethernet, WI-FI 802.11 b/g, RS-485, Ethernet UDP e até USB, com a utilização de um Computador Pessoal. Cada um tem sua aplicação e o que difere um do outro é a distância de comunicação, taxa de dados (velocidade) e segurança de dados (SILVA, 2013). A Figura 2-13 mostra o módulo RN-171 produzido pela empresa Microchip que possui todas as funcionalidades para comunicação via WI-FI.

Figura 2-13- Módulo para comunicação via WI-FI.
Fonte: MICROCHIP, 2013.

O módulo RN-171 tem baixo consumo de energia, e possui embutido o módulo TCP/IP completo para 802.11b/g. Seu tamanho é de 27x18x3,1 milímetros, o que o torna perfeito para aplicações móveis sem fio, como monitoramento de ativos, sensores e dispositivos portáteis a pilha (MICROCHIP, 2014).

2.2.3.3 RÁDIO FREQUÊNCY

Frequência de rádio, ou ondas de radiofrequência se refere a uma taxa de oscilação no intervalo entre 3KHz e 300 GHz. Essas ondas são campos magnéticos utilizados nas comunicações sem fio. A comunicação se dá com o transporte de energia de um ponto a outro sem a necessidade de fios, como nas transmissões de televisão, rádio e celulares. O sinal se propaga no ar em forma de ondas ou em linha reta (SENAI, 2014).

Na plataforma Arduino, a transmissão de dados via RF é uma alternativa eficiente e barata. Com um Kit de módulo de RF Transmissor e um Receptor 433 MHz, é possível enviar ou receber dados sem necessidade de fios.

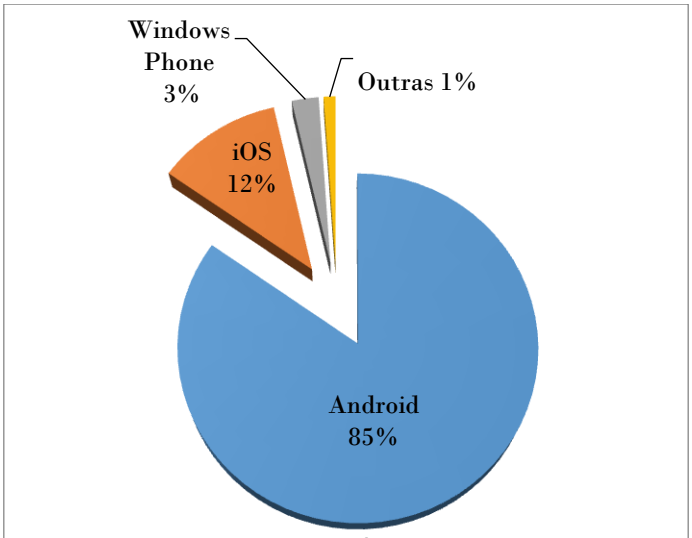
Figura 2-14 - Kit APC 220.
Fonte: (DX, 2014).

O módulo APC220 possui um microprocessador de alto desempenho . É capaz de transmitir a uma distancia de até 1000m (linha de visão) a 9600 bips, o que o torna perfeito para esse projeto.

2.3 ANDROID

O Android se refere à plataforma desenvolvida especialmente para dispositivos móveis como os *Smartphone's* e *tablets.* É composta por sistema operacional, *middleware's* e um conjunto de aplicativos disponíveis para o usuário como agenda de contatos, navegador para internet, GPS, aplicativos para mídias diversas, e claro, o próprio telefone (MONTEIRO, 2012).

O Android conquistou seu espaço de destaque no mercado mundial à frente de plataformas como *iOS, Symbian, Windows Phone* e outros. De acordo com os dados do relatório referente ao segundo trimestre de 2014 do IDC (*International Data Corporation*) a plataforma Android atinge 85% da quota de mercado (IDC, 2014). A Figura 2-15 mostra o gráfico que representa os dados de mercado das plataformas móveis mais populares.



Fonte dos dados: (IDC, 2014).

Figura 2-15 - Participação no Mercado.

A popularidade do Android continua em crescimento não só pela quantidade de dispositivos produzidos, mas principalmente pela facilidade para desenvolvimento de aplicativos e por possuir uma API rica favorecendo o acesso à diversos recursos de hardware, como WI-FI e GPS (MONTEIRO, 2012).

Para os desenvolvedores Android é disponibilizado kit de desenvolvimento de software (SDK), bem como a loja de distribuição de aplicativos(Google Play), o que viabiliza o mercado dessa área de forma simples, prática e com baixo custo. Toda essa facilidade não só atrai novos desenvolvedores a cada dia como também faz com que maior quantidade de pessoas tenham acesso aos aplicativos oferecidos (MONTEIRO, 2012).

2.3.1 HISTÓRIA

A história do Android começa em 2003, com grandes marcos para a empresa que depois de muitas melhorias durante os anos seguintes, lançou a versão 4.4 em outubro de 2013 (XCUBE LABS, 2014). O resumo de toda jornada no desenvolvimento da plataforma pode ser visualizado na Figura 2-16.

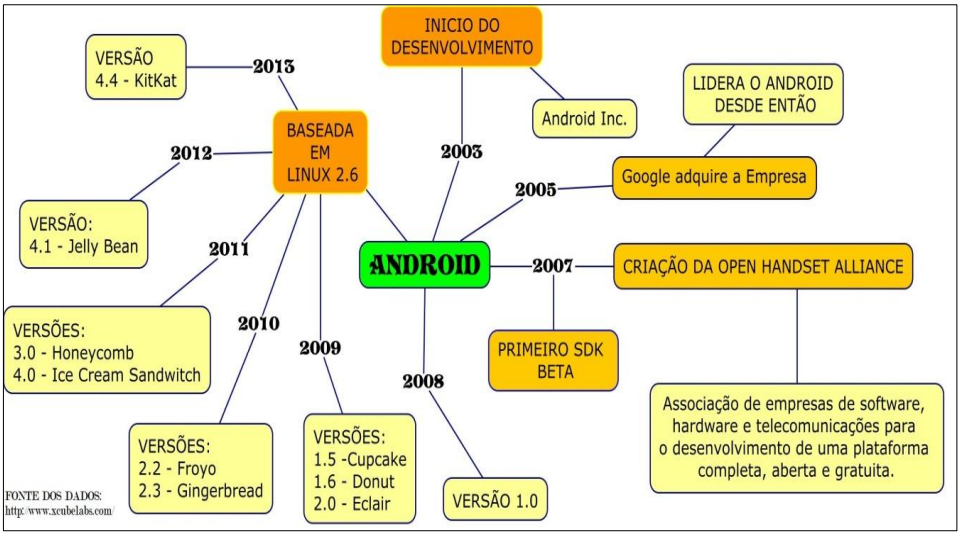


Figura 2-16 - Android em anos.
Fonte: Elaborada pela Autora.

O projeto Android é open source distribuído sob licença Apache 2.0, o que significa que o desenvolvedor pode ter acesso aos códigos-fonte caso queira contribuir para o projeto.

2.3.2 ARQUITETURA ANDROID

O Android é um sistema formado por uma pilha de softwares de diferentes camadas. Cada camada é composta por vários programas com funções específicas do sistema operacional (ANDROID DEVELOPERS, 2014). A Figura 2-17 apresenta uma amostra do diagrama dessa arquitetura.

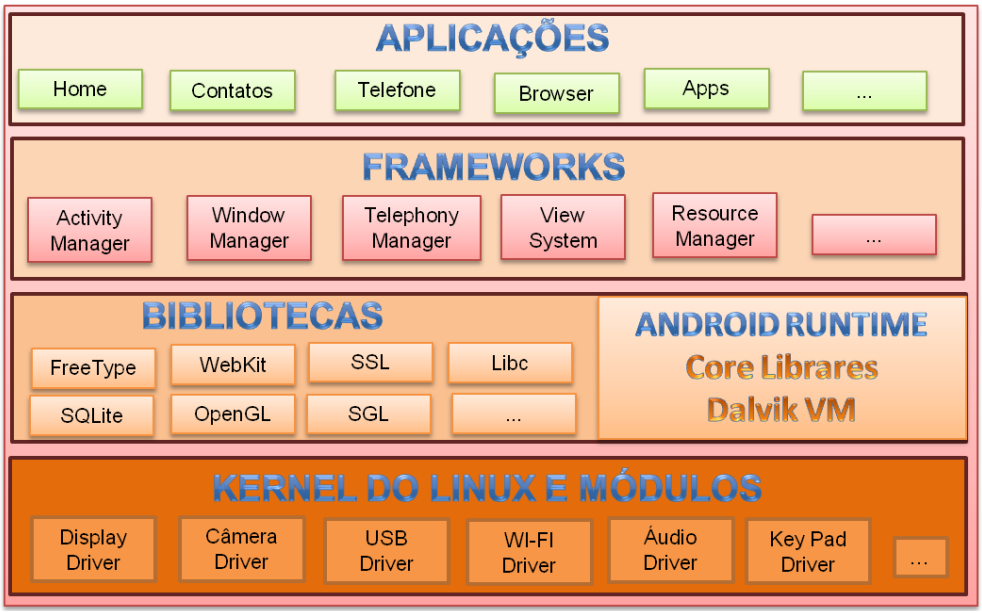


Figura 2-17 - Arquitetura Android.
Fonte: Elaborada pela Autora.

O sistema operacional Android é construído sobre o Kernel do Linux 2.6, que atua como camada de abstração entre o hardware e outras camadas de software e está presente em todas as funcionalidades básicas do Android, como gerenciamento de memória, gerenciamento de processos, redes, configuração de segurança entre outras (ANDROID DEVELOPERS, 2014).

As bibliotecas nativas do Android, que compõem a segunda camada da arquitetura são escritas em C ou C++ e cada uma é específica para um determinado hardware. Na segunda camada também se encontram a máquina Virtual *Dalvik* (JVM) e o Core Java *Libraries* (ANDROID DEVELOPERS, 2014).

A terceira camada possui frameworks que interagem diretamente com as aplicações android, como a gestão das funções básicas do telefone, gestão de recursos, chamadas de voz, entre outras. A quarta camada se refere às aplicações, ou seja, a camada de interação com usuário (ANDROID DEVELOPERS, 2014).

2.3.3 ESTRUTURA DE UMA APLICAÇÃO ANDROID

Apesar do desenvolvimento Android ser na linguagem Java, a estrutura da aplicação segue um formato diferenciado. Segundo (SPINOLA, 2014), a aplicação Android possui 4 elementos essenciais:

- ***Activity*** - Classe responsável pela interface com o usuário.
- ***Intents*** - Usado para que a aplicação reaja a algum evento externo
- ***Service*** - Código executado durante toda a aplicação(Ex: *Thread*).
- ***Content Provider***- Classe que permite o compartilhamento de dados entre aplicações.

Uma aplicação android não precisa ter os quatro elementos, mas sempre será necessário a combinação de dois ou mais desses elementos para que o código funcione.(SPINOLA, 2014).

2.3.3.1 CICLO DE VIDA DE UMA ACTIVITY

As atividades de uma aplicação Android representam classes que são executadas no ato de seu chamado. O inicio do ciclo de vida dessas atividades

começam a partir do momento da criação de uma aplicação até o seu término (ANDERSON et. al., 2014). A Figura 2-18 apresenta o ciclo de vida de atividade de uma aplicação Android.

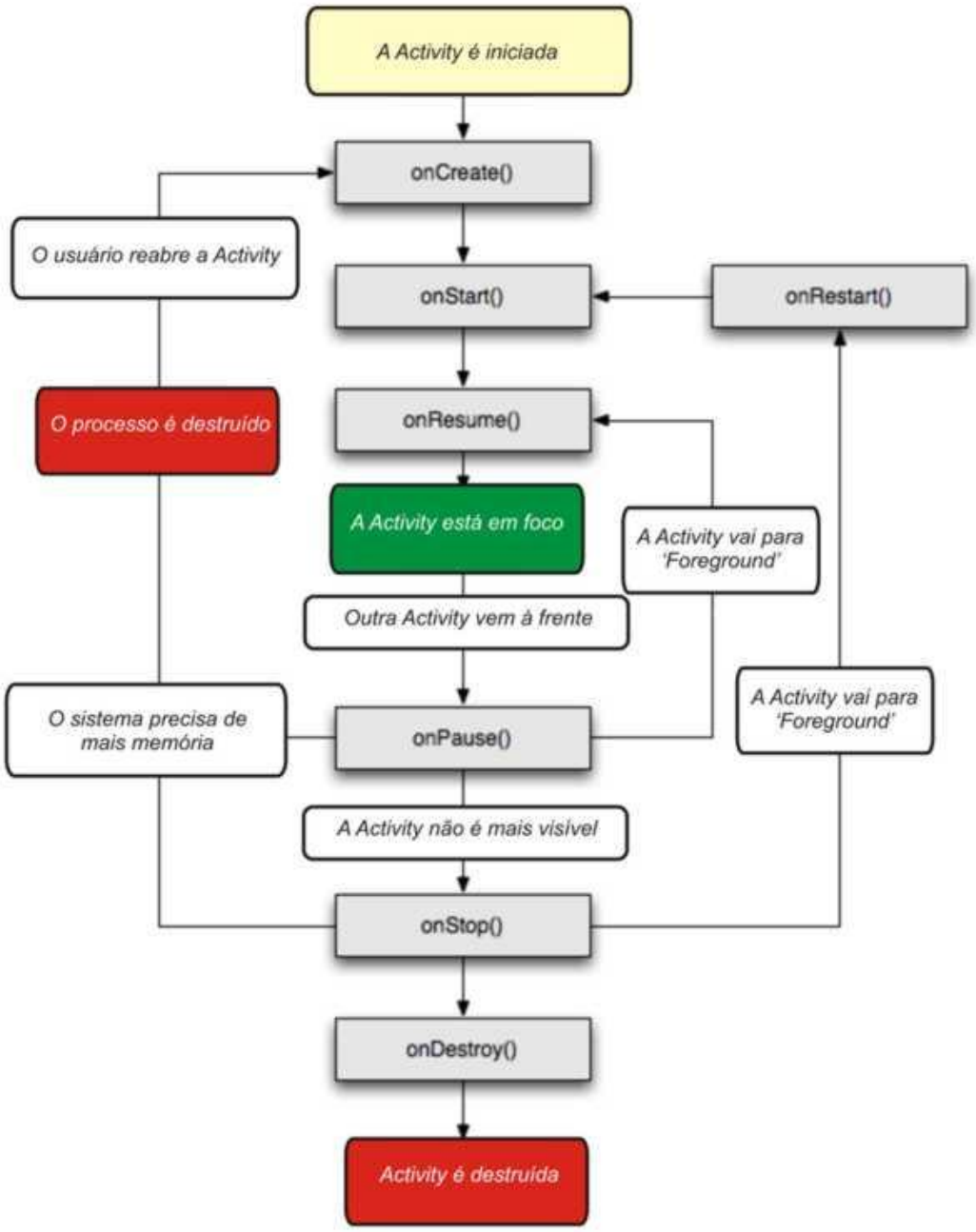


Figura 2-18 - Ciclo de vida de uma activity.
Fonte: NERY, 2014.

Todas as *activitys* da aplicação devem ser devidamente informadas no arquivo *AndroidManifest.xml*. Trata-se de um arquivo de extrema importância para o projeto Android e fica localizado na sua raiz. É no Arquivo do *manisfest* que partes importantes do projeto são descritas (ANDERSON; AUGUSTO, 2014).

3. METODOLOGIA

Como já citado no trabalho, o termo VANT se refere a todo tipo de aeronave que não necessite de um piloto embarcado. São aeronaves ideais para realização de atividades exaustivas para um ser humano ou exercidas em locais de difícil acesso. O modelo definido para esse projeto é o quadrotor - uma aeronave dotada de quatro motores e dois pares de hélices que são programados para girarem no sentido inverso uma da outra.

A pesquisa teve início com estudos sobre o VANT e suas diversas aplicabilidades. Essa parcela do trabalho foi realizada por meio de leitura à artigos relacionados ao assunto e trabalhos acadêmicos concluídos ou em andamento. A importância desse estudo está relacionada em agregar conhecimento preliminar sobre o objeto (VANT) que será manipulado pelo software desenvolvido.

Para se chegar às variáveis de controle do VANT foi necessário obter-se informações e esclarecimentos sobre diversos assuntos associados ao funcionamento, manipulação, navegação e controle de aeronaves com base em publicações sobre o tema e discussões na internet em grupos específicos de robótica e afins. Apesar do foco do trabalho ser a implementação de um software de controle, não seria possível desenvolver um sistema sem um prévio conhecimento do que realmente se trata um VANT.

Alem do estudo particular sobre a aerodinâmica do VANT, foi necessário determinar o meio real de comunicação entre o Sistema Operacional Android e a plataforma Arduino, base de construção de protótipos.

Para atender as condições do problema exposto, foi realizado atividade de levantamento de requisitos como parte fundamental para o projeto, o que possibilitou a compreensão do problema e o que seria definido para resolvê-lo. Essa atividade engloba a busca por uma série de informações que vão desde o que é indispensável para o desenvolvimento do software até o que é necessário para o usuário final do sistema.

Após definido o material necessário e os requisitos mínimos para o sistema, foi iniciado estudo sobre a plataforma para desenvolvimento do software, o Android. Paralelamente foi realizado uma analise minuciosa sobre a utilização de bibliotecas e protocolos de redes indispensáveis para o projeto. Um dos pontos notórios para desenvolvimento do controle foi a API de serviços do Google Play que oferece uma biblioteca com vários serviços essenciais para a implementação, garantindo o perfeito desempenho do software para Android. Entre os serviços oferecidos estão: autenticação para os serviços Google, acesso a configuração de privacidade do usuário com base na localização, aprimoramento da experiência com aplicativos e permissão de pesquisas off-line.

Além do sistema operacional, foi necessário decidir a IDE (*Integrated Development Environment*) para desenvolvimento do software em questão. A escolha pelo Eclipse se deu por se tratar de um ambiente favorável que auxilia no desenvolvimento e possui suporte a várias linguagens a partir de *plugins*. O Eclipse faz parte do SDK (*Software Development Kit*) recomendado para desenvolvedores Android, é *open source* e é implementado em Java.

Resolvido questões referentes a Sistema Operacional e Plataforma de Desenvolvimento para o *software*, o próximo passo coube focar esforços à Plataforma Arduino. No projeto, a Plataforma Arduino foi a base para criação do protótipo em que se realizaram todos os testes com o software. Para isso foi inevitável o envolvimento com robótica partindo das partes mais simples, como acender um LED, por exemplo, até atividades mais complexas, que envolveram conhecimento mais amplo, tanto na montagem do protótipo como no sistema desenvolvido para manipulação do hardware em conversa com um ambiente externo à IDE Arduino. Para a criação do protótipo utilizado nos testes foi necessário a aquisição dos itens descritos no Quadro 6.

Quadro 6 - Itens utilizados no projeto.

QUADRO VI	
NOME	**DESCRIÇÃO**
Microcontrolador Arduino UNO	Plataforma utilizada para construção de protótipos
APC220	Comunicador de Radiofrequência
Tablet com *hos*t-USB	Entrada USB com especificação adequada para suportar conexões com dispositivos externos

Cabo USB-OTG	Cabo com função *On The Go* que permite conexão com dispositivos externos
Acessórios para prototipagem em Arduino: LED, Jumper, Buzzer, Display digital de sete segmentos, servo motor e protoboard	

A implementação dos códigos para a plataforma Arduino foi feita na IDE Arduino, com linguagem de programação c++ de acordo com as necessidades de experimentos para o projeto.

O ponto central da pesquisa foi a comunicação entre o sistema operacional Android e a plataforma Arduino. Para atender aos requisitos de comunicação entre o VANT e o dispositivo móvel, um dos principais estudos esteve relacionado aos protocolos de rede sem fio. Foi necessário levar em consideração o tipo de dispositivo utilizado, os módulos acoplados ao próprio VANT e o alcance do sinal que o dispositivo de rede pode alcançar.

Os protocolos de rede que fizeram parte do estudo foram:

- Padrão WI-FI - 802.11;
- Protocolo *Bluetooth* - 802.15;
- Radio *Frenquency* - (RF) ;

Depois de iniciar a pesquisa sobre comunicação de redes, os estudos e consequentemente os testes, foram realizados por partes:

1. Comunicação direta entre o computador desktop e plataforma Arduino via cabo USB;
2. Maquina simulada do Android com finalidade de virtualizar a comunicação de rede e realizar uma comunicação direta entre o sistema Android Simulado e a plataforma Arduino via cabo USB, quando houve falta de itens que permitissem testar o sistema.
3. Estudos teóricos para as formas de comunicação de rede sem fio;
4. Instalação, calibragem e testes com o módulo de radiofrequência.
5. Testes de comunicação via radiofrequência.

Um fluxograma do processo de testes para desktop e em seguida os testes

em android é apresentado na Figura 3-1.

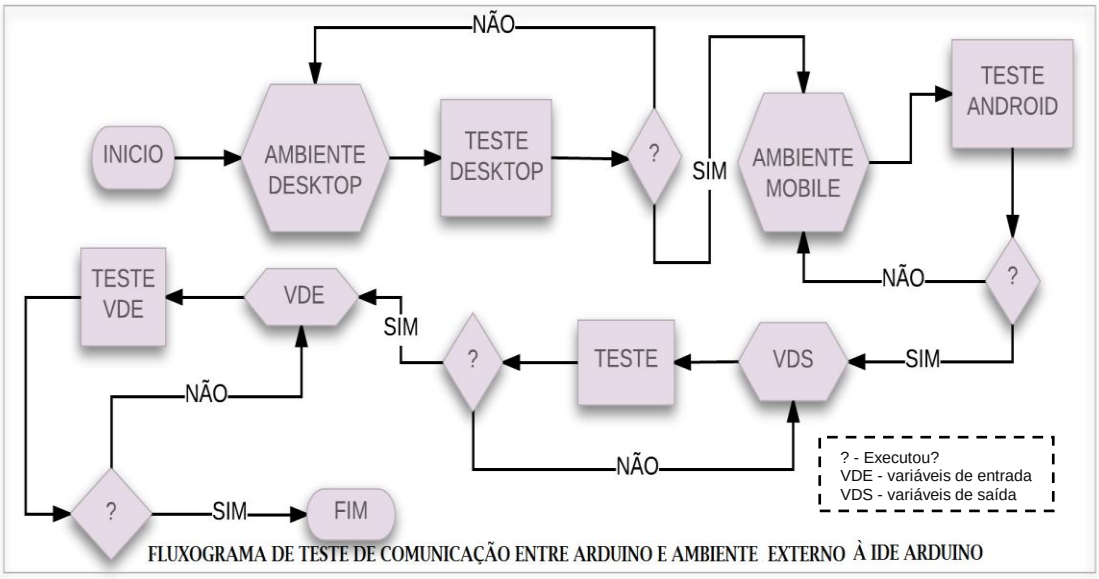


Figura 3-1 - Fluxograma de Teste de Comunicação da Plataforma Arduino com Ambiente Externo à IDE Arduino.
Fonte: Elaborada pela Autora.

Após definir todos os pontos indispensáveis do projeto para a implementação do controle remoto foi contemplado a escolha de um modelo de interface para o usuário. Como base em estudo de outros trabalhos sobre o tema, ficou definido que a melhor interface para o controle seria o Joystick, que simula um controle para jogos.

Para implementação dessa interface foi realizado estudo de um projeto para biblioteca que traz definida todas as configurações necessárias para atender o bom funcionamento de um controle joystick. A Figura 3-2 mostra o desenho de um controle não configurado, conforme é proposto no projeto.

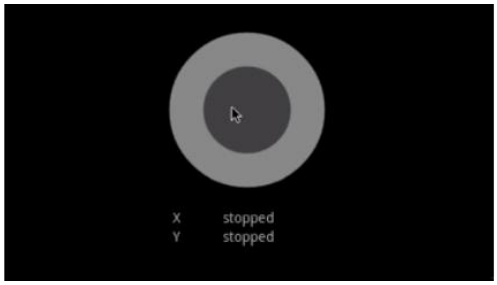


Figura 3-2 - Exemplo do controle joystick do projeto *mobile-anarchy-widgets.*
Fonte: MOBILE-ANARCHY*, 2013*.

O projeto para o Joystick é composto por classes que simulam um controle joystick na tela e proporciona uma interface que facilita o movimento do controle

com base em cálculos matemáticos em todos os ângulos de movimento como se verificou com sua utilização.

Foram utilizadas ferramentas para compor o texto do projeto, entre elas estão: Lucidchart , para criação de esquemas e fluxogramas; Frtizing, para criação dos esquemas de protótipo de hardware; Cmap, para composição de mapas mentais; e Ferramenta de captura para capturar a tela do aplicativo em android.

4. DESENVOLVIMENTO DO SISTEMA

O objetivo do trabalho estava em possibilitar o controle de um VANT do tipo mini quadrotor com base na plataforma Arduino, por meio de um dispositivo móvel, em ambientes de redes sem fio. Para obtenção de resultados foi necessário trabalhar em estudos de quatro pontos fundamentais nesta pesquisa: o VANT, a plataforma Arduino, comunicação de rede sem fio e a tecnologia Android.

Nesse capítulo é apresentado os detalhes da montagem dos protótipos utilizados, desenvolvimento do software para o protótipo e desenvolvimento do aplicativo para o controle. Para atender ao objetivo do projeto, fez-se necessário a atividade de levantamento de requisitos, que estão apresentados no Quadro 7.

Quadro 7 - Requisitos para o Sistema de Controle Remoto.

QUADRO VII	
NOME	**DESCRIÇÃO**
REQUISITOS FUNCIONAIS	Opção de conectar e desconectar
	Enviar comandos para a aeronave
	Operar manualmente a aeronave através de dois controles joysticks
REQUSITOS NÃO FUNCIONAIS	Sistema Operacional – Android versão mínima 3.2.
	Conexão USB-OTG e Modo *host*-USB .
	IDE(*Integrated Development Enviroment*) – Eclipse com Android SDK – API mínima 14.
	RF *Magic* ou RF-ANET e *Driver* para RF.
	Arduino IDE 1.0.5.
	Linguagens – Java e c++.

O sistema possui duas partes distintas: hardware e software, como pode ser visto no esquema apresentado na Figura 4-1.

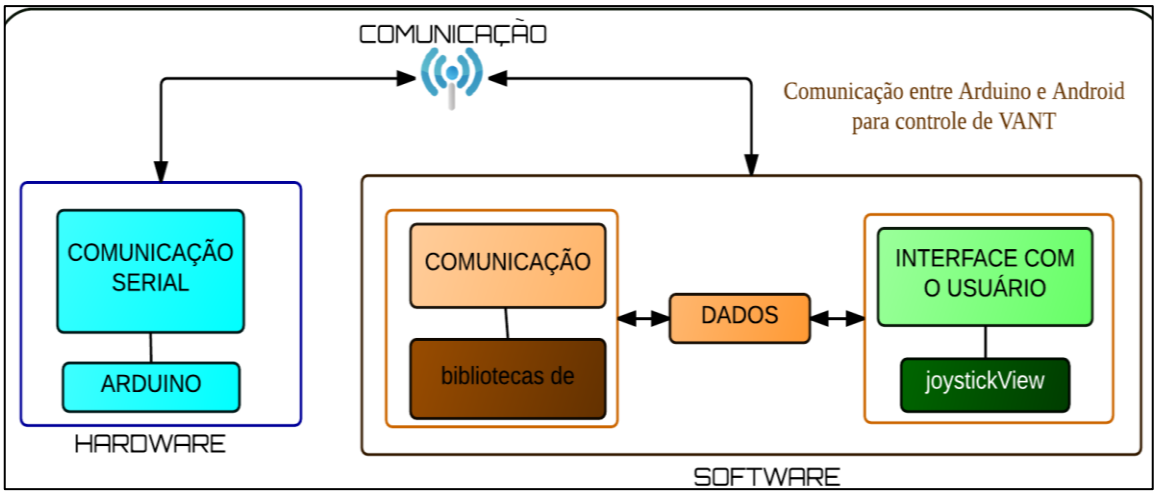


Figura 4-1 - Comunicação entre Android e Arduino para Sistema de Controle do VANT.
Fonte: Elaborada pela Autora.

A importância da criação de protótipos de hardware está no fato de que os testes realizados com o sistema em desenvolvimento não poderiam ser aplicados diretamente com um VANT real, visto o risco de acidentes ou prejuízos causados por falhas. Para o protótipo também foi desenvolvido o software que permite a conversa com o aplicativo de controle. Os passos e detalhamento dessas partes são apresentados nas próximas subseções.

4.1 HARDWARE - CRIAÇÃO DE PROTÓTIPO DE HARDWARE

O protótipo de hardware para testes dos comandos que seriam recebidos da aplicação do dispositivo móvel, possuem funções básicas para simular entrada e saída de dados pela porta serial na plataforma Arduino. Para esse projeto foi utilizada a placa microcontroladora Arduino UNO, apesar de que a ideal para simulações seria a placa Crius 1.0 ou similar, que pode ser usada para montagem de drones.

Foi necessário pensar em alguns modelos e testar vários componentes com a finalidade de alcançar o melhor resultado na simulação dos estados que representam os movimentos do VANT do tipo quadrotor. O protótipo final é um projeto para ligar e desligar um LED, movimentar um servo motor e acionar um Buzzer por meio da porta serial. Para cada comando recebido um valor é apresentado no display, o que confirma que o comando foi enviado e recebido. A Figura 4-2 mostra o protótipo do hardware.

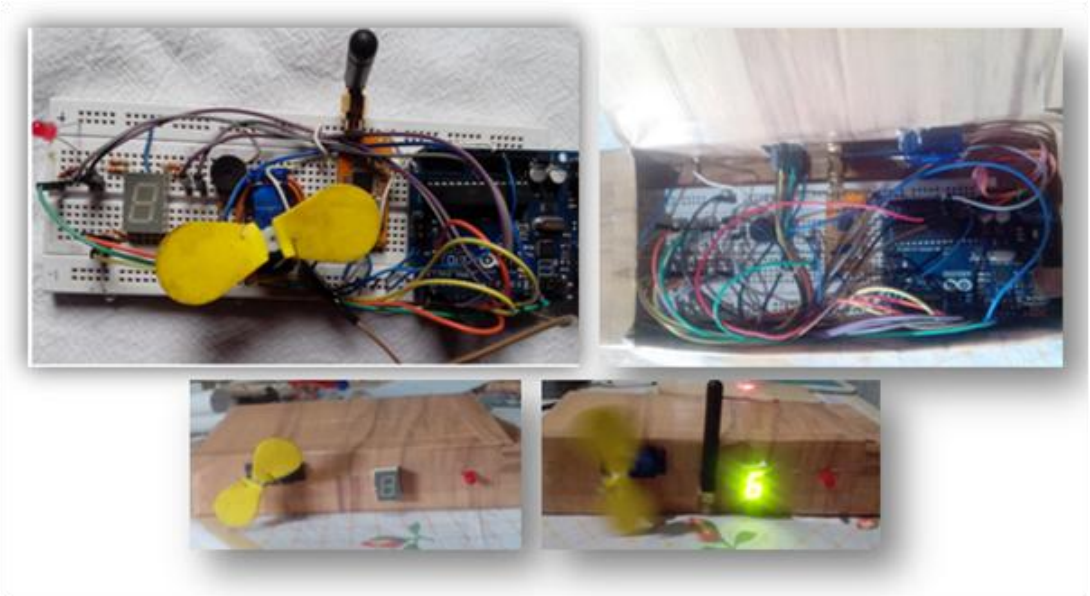


Figura 4-2 - Protótipo em Arduino.

Os itens utilizados na criação desse protótipo foram:

- Placa microcontroladora Arduino UNO - alimentação à energia.
- *Jumper* - Fios contendo parte metálica utilizado para ligação dos circuitos eletrônicos.
- *Protoboard* - Placa de ensaio com furos e conexões condutoras utilizada para circuitos elétricos experimentais.
- LED (*Light Emitting Diode*) - Quando energizado emite luz visível. No projeto é utilizado para receber comandos dos dispositivos externos ao ambiente de desenvolvimento Arduino. O LED possui um lado positivo e um negativo. O lado positivo deve ser ligado à uma resistência para controle da corrente elétrica.

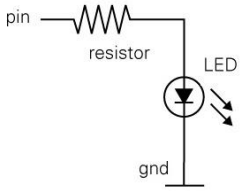


Figura 4-3 - Esquema para um LED.

- Resistência 330 ohms - Utilizada para controlar a corrente de energia para o Display.
- Resistência 220 ohms -Utilizada para controlar a corrente de energia para o LED.
- Resistência 150 ohms - Utilizada para controlar a corrente de energia para o Buzzer.

- APC 220 - Módulo de rádio para comunicação por radiofrequência.
- BUZZER - Buzina para emitir sons, capaz de controlar a frequência por meio de código. A função para acionar o *Buzzer* traz 3 parâmetros: ton(10, 100, 1000). O primeiro parâmetro corresponde ao pino definido para o *Buzzer.* O segundo, é a frequência, ou seja, quanto maior o valor, mais agudo é o som. O terceiro, é o tempo de duração em milissegundos.
- Servo Motor - Pequena máquina que apresenta movimento conforme o comando que é recebido. Possui liberdade de movimento de apenas 180º, e são precisos quanto a posição, o que permite melhor manipulação pelo controle. As ligações são feitas em positivo (5v), GND e um pino de saída.
- Display Digital de 7 segmentos - Display que apresenta um numero ou letra codificado por 7 segmentos. Para cada segmento e mais um que representa o ponto, é utilizado um resistor de 330 ohms para controle de energia.

O esquema do protótipo do hardware que recebe comandos do controle é apresentado na Figura 4-4.

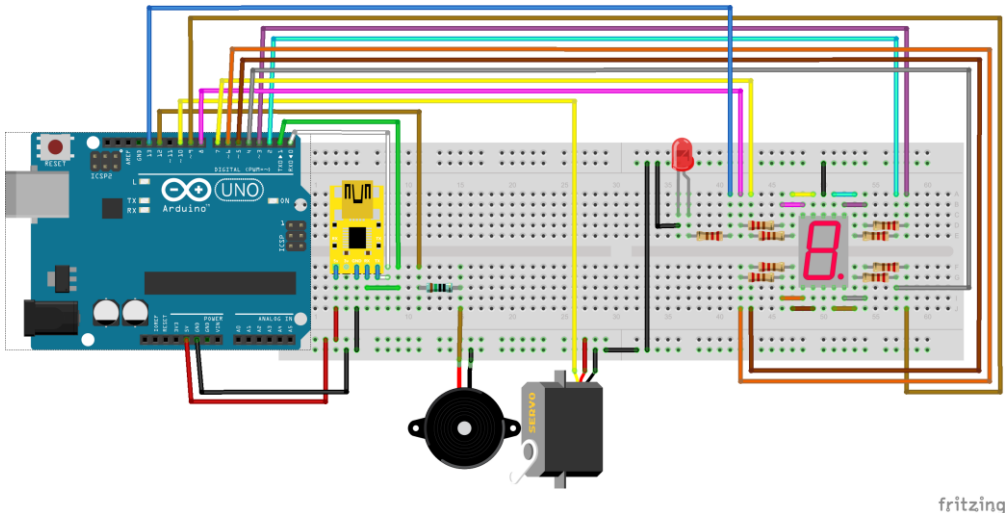


Figura 4-4 - Protótipo Para Teste de Comunicação via Porta Serial.

Nota-se que foram utilizados somente recursos suficientes que garantissem a sincronização externa entre a plataforma Arduino e um dispositivo externo ao IDE Arduino.

O código embarcado no protótipo permite a entrada de dados pela porta serial configurada com *baud rate* 9600, que é a taxa definida para transmissão padrão. Esse mesmo valor está configurado nas placas APC220. Os comandos recebidos pela porta serial do protótipo possibilitam suas ações e seguem o protocolo apresentado no Quadro 8:

Quadro 8 - Comandos para Ação na Arduino.

QUADRO VIII		
COMANDO	**VALOR**	**AÇÃO**
f	1	Acender o LED
g	2	Desligar o LED
d	3	Acionar *Buzzer* frequência agudo
e	4	Acionar *Buzzer* frequência média
a	5	Gira o ServoMotor para direita
b	6	Gira o ServoMotor para esquerda
c	7	Gira o ServoMotor lento
*****	8	Mostra o valor da ação no Display

Para testar os valores diretamente no protótipo pode ser utilizado o monitor serial do IDE Arduino com a entrada dos comandos.

A plataforma Arduino disponibiliza uma série de bibliotecas com exemplos na própria IDE, que permitem formas variadas de manipulação de dados. Durante a montagem e definição para o protótipo foram testadas várias bibliotecas com a finalidade de estabelecer a comunicação no caminho de ida e volta por meio de radiofrequência com dispositivo móvel. As limitações do módulo de rádio APC220 não permitem a comunicação paralela síncrona (Receber e enviar dados ao mesmo tempo), conforme disponível nos exemplos da IDE para comunicação direta entre duas placas Arduino.

Uma das bibliotecas que permite a escrita e leitura em duas portas seriais ao mesmo tempo, é a biblioteca *NewsoftSerial*, que possibilita criar a comunicação em novas portas alem da 0 e 1, que correspondem a RX TX respectivamente. O problema é que essa biblioteca funciona bem para comunicar duas placas Arduino conectadas por *jumpers*, como mostra o esquema na Figura 4-5.

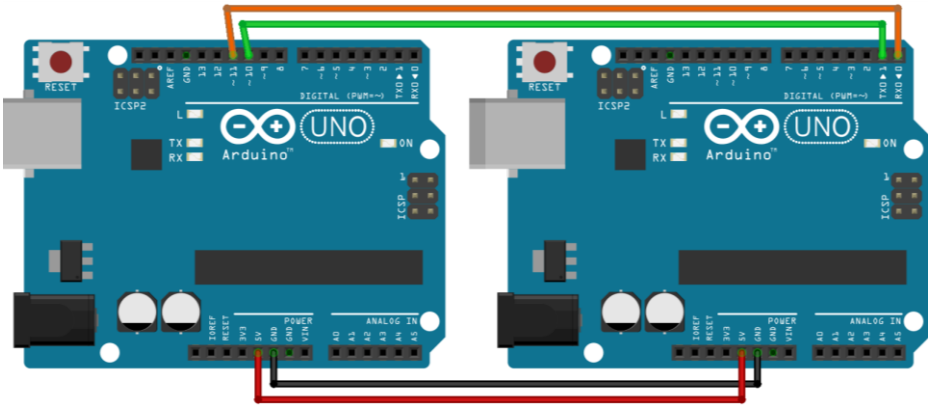


Figura 4-5 - Exemplo para uso do NewsoftSerial.
Fonte: Elaborado pela Autora.

O exemplo da biblioteca *NewsoftSerial* traz a pinagem 10 para RX e 11 para TX, e permite a configuração de *baud rate* em até 115200 diretamente na placa. Apesar da possibilidade de criar várias instancias em quase todos pinos digitais disponíveis e configuráveis, apenas um pino pode ficar ativo por vez, por isso, apesar de criar o caminhos dos dois lados, para transmitir e receber dados simultaneamente é necessário manipulação de pacotes com tempo de espera definidos em estados de recepção e transmissão.

Outras bibliotecas mais eficientes para comunicação paralela e síncrona possuem restrições para o uso do Arduino UNO, que fora utilizado para o projeto. Na maioria dos exemplos, são recomendados as placas Arduino Leonardo ou Arduino Mega, que possuem uma quantidade bem maior de pinos que a placa Arduino UNO.

Como o hardware Arduino possui suas próprias limitações e o projeto visa a comunicação via radiofrequência, a comunicação foi estabelecida com as funções nativas do Arduino. As funções utilizadas para comunicação serial da placa Arduino foram:

- **Serial.begin(*baud rate*)** - Essencial para comunicação serial, pois configura a taxa de transmissão em bits por segundo.
- **Serial.available()** - Verifica os dados para leitura da porta serial e retorna a quantidade de bytes disponíveis no buffer de leitura.
- **Serial.read() -** Lê o ultimo dado no buffer de entrada serial.
- **Serial.print()** - Escreve um dado na porta serial conforme parâmetro.
- **Serial.write()** - Escreve um byte na porta serial.

A comunicação via radiofrequência acontece em baixo nível e não é notada pelo usuário, ou seja, é a mesma forma que ocorre a comunicação via USB . Só depois de feito o *upload* do software para o hardware Arduino é que se conecta os pontos RX e TX da placa APC220 nas portas TX e RX respectivamente, conforme exemplifica a Figura 4-6.

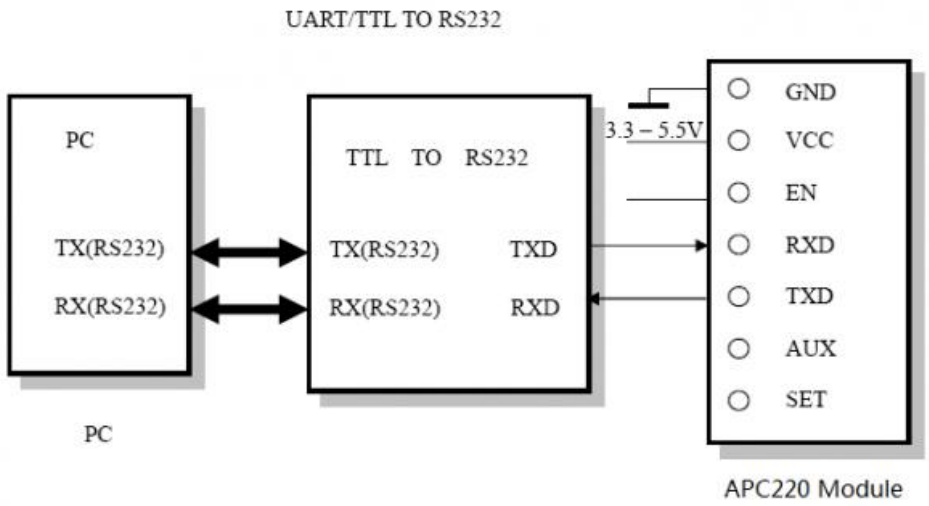


Figura 4-6 - Esquema para comunicação Apc220.
Fonte: Manual APC220, 2014.

A comunicação de radiofrequência somente acontecerá se as duas placas de rádio, no caso o conjunto de APC220, estiverem dentro da mesma calibragem de frequência. Para isso, primeiro foi imprescindível verificar a documentação da placa de rádio que mostra todo o caminho para garantir seu perfeito funcionamento.

De acordo com instruções da documentação, o *driver* da placa deve estar instalado na maquina para realizar a calibragem. Alguns computadores realizam a instalação do *driver* automaticamente assim que reconhecem a entrada pela conexão USB. Quando isso não acontece, o trabalho é feito manualmente e, nesse caso é importante saber qual chip está acoplado no adaptador USB.

Para esse projeto, foi utilizado o adaptador USB com chip CP2102, que vem como acessório padrão da placa APC220. O drive para o chip é disponibilizado pela *Silicon Laboratories, e* está disponível para baixar no site da documentação da placa APC220 *(*MANUAL APC220*,* 2014*).*

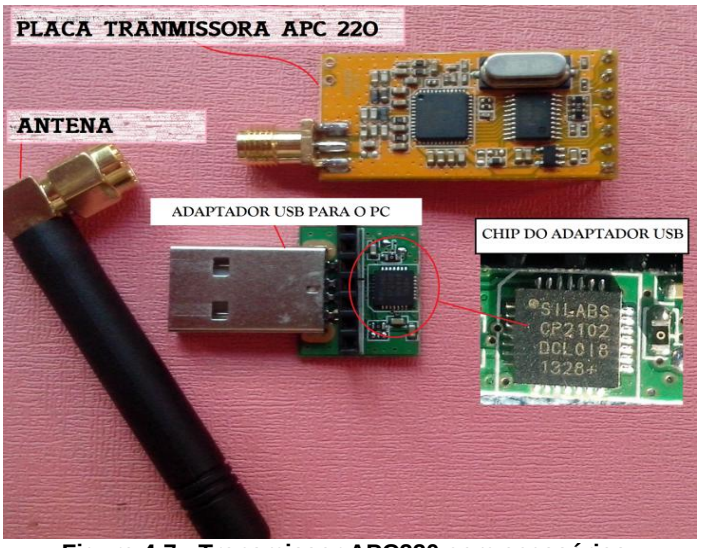


Figura 4-7 - Transmissor APC220 com acessórios.
Fonte: Elaborada pela Autora.

Na Figura 4-7 pode observar a numeração do chip que aparece junto ao adaptador USB. Após a instalação do *driver é* necessário verificar qual porta USB está alocada, o que pode ser averiguado em "gerenciamento de dispositivos" do computador.

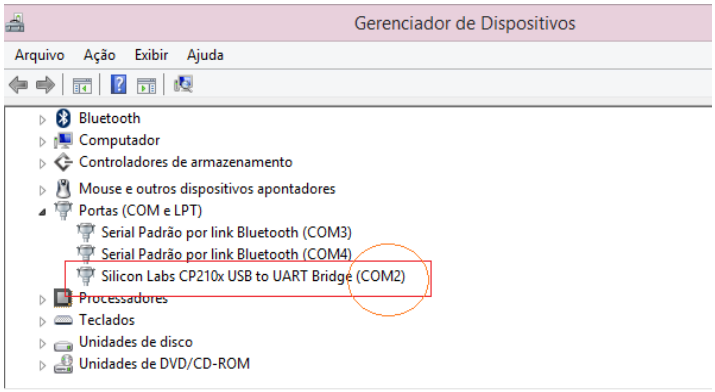


Figura 4-8 - Localização da porta de entrada do driver.

O próximo passo para a comunicação é baixar e instalar o programa que faz a calibragem de frequência. Para esse projeto, foi utilizado o software RF-Magic 1.2, indicado no manual da placa, com a finalidade de calibrar a frequência e ajustes necessários para os testes. Para evitar erros no sistema é recomendado conectar o adaptador na aporta USB do computador, certificar de que o *driver* foi reconhecido e só depois executar o programa *RF-Magic* sempre no modo administrador.

Um ponto interessante que surgiu durante as configurações para calibragem de frequência, foi que muitas vezes o computador não reconhecia a

entrada USB e o software de configuração não respondia, ou apresentava mensagens de erro. Isso ocorreu devido a geração aleatória de porta, quando as portas já estavam sendo utilizadas e logo ocorria a sobrecarga. A solução encontrada para esse problema foi fazer alteração da porta diretamente no *driver,* em gerenciamento de dispositivos, local em que é possível a modificação da porta padrão para alguma outra que esteja disponível no sistema.

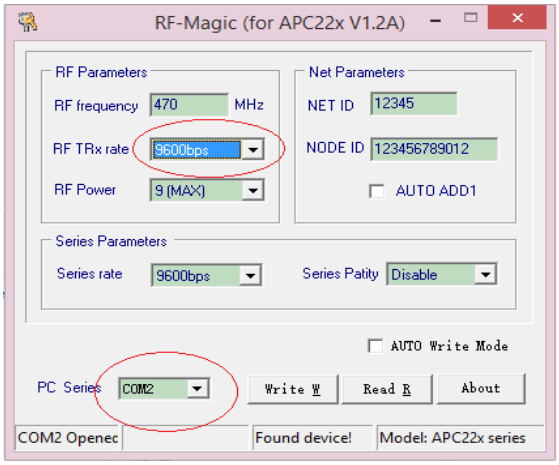


Figura 4-9 - Janela do software RF-Magic.

As principais configurações aqui são a frequência, taxa de dados(*baud rate*) e taxa de RF-TRx que precisam ser iguais nas duas placas e nos programas de testes. A placa APC220 permite que a taxa RF TRx possa ser configurada somente até 19200. Já a *Series rate*, vai até 57600. Para esse projeto foi utilizado a configuração padrão em 9600 tanto para taxa RF TRx como para *Series rate*, apesar de que para estabelecer uma comunicação direta entre o Android e a placa de rádio e receber dados do microcontrolador, seria necessário pelo menos a configuração 57600 para Series rate.

A taxa de dados (Series rate) define a velocidade de dados por segundo para transmissão pela porta serial. Entre o hardware Arduino UNO e a IDE Arduino, a comunicação acontece naturalmente em qualquer uma das taxas definidas, que podem ser: 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, e 115200. Quando se utiliza a telemetria por radiofrequência, essa taxa alem de padronizada entre a configuração das placas e o software do Arduino, exige também estabelecer uma velocidade mínima que garanta a transmissão sem perda total de dados.

Essa questão de perda de dados foi manifestada em vários aspectos e situações durante o projeto. Um dos pontos mais críticos, é o fato de que o sistema android possui limitações quanto a conversa com periféricos. O objetivo era enviar comandos para um VANT de forma a controlá-lo, e com *delays* de mais de 10 segundos seria inviável.

Em busca da solução para esse problema, foi observado que em trabalhos que abordam temas semelhantes, utilizou-se sempre de um outro meio de comunicação para chegar às variáveis do VANT. Geralmente é utilizado um módulo Bluetooth, virtualização do VANT, módulo Wi-Fi ou mesmo drivers que permitam essa comunicação direta. Uma das soluções mais comuns é a utilização de *Shields* para ampliar a capacidade do Arduino.

Com base nessas experiências, a solução encontrada foi criar uma ponte com outro protótipo que acopla a placa de radiofrequência para envio de dados ao protótipo principal. O Esquema desse protótipo é apresentado na Figura 4-10.

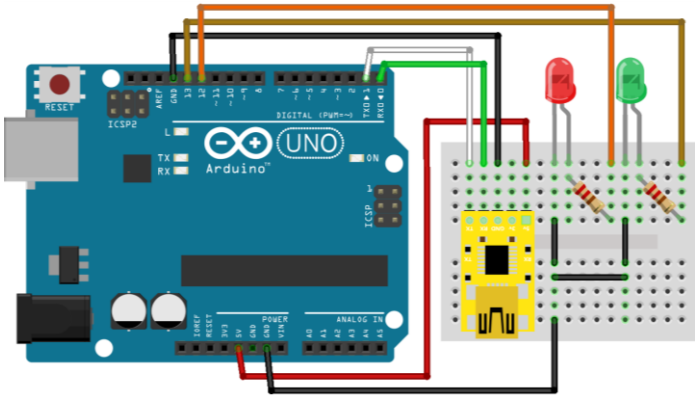


Figura 4-10 - Esquema do protótipo Ponte.

A função desse protótipo é estabelecer uma conversa com o protótipo principal. No código fonte embarcado na placa só existe a abertura da porta serial com *baud rate* 9600, básica para comunicação entre as duas plataformas Arduinos. Tudo que for escrito na porta serial do arquivo ponte é transferido para o arquivo principal, com *delay* mínimo, o que atende a necessidade de comandos precisos, no caso do controle de um VANT.

Figura 4-11 - Protótipo ponte.

O protótipo ponte funciona como um rádio que vai receber os dados e enviá-los por meio de radiofrequência para o protótipo principal. Para isso, estará acoplado ao Tablet e conectado via cabo USB para receber os comandos do aplicativo em Android.

4.2 O SOFTWARE

Para o desenvolvimento do sistema de controle foi necessário definir etapas e procedimentos que tinham a finalidade de chegar ao aplicativo final. A primeira etapa se refere ao sistema de comunicação entre uma aplicação desktop e Arduino. E a segunda etapa, ao desenvolvimento do aplicativo Android.

4.2.1 SISTEMA DESKTOP

O objetivo do sistema era estabelecer uma conversa entre o Arduino e um sistema externo ao ambiente de desenvolvimento IDE Arduino. Para o procedimento dessa etapa, foi utilizado a API RXTX baseada na API Javacomm, que, segundo (SILVA, 2014) possibilita a comunicação via USB, comunicação paralela síncrona, ou comunicação serial, entre uma aplicação Java e um hardware externo. Teoria que pôde ser averiguada e testada com o software desenvolvido.

A aplicação, quando iniciada verifica a conexão entre as partes e retorna feedback no console da maquina que diz se a conexão foi estabelecida. Em caso positivo, a aplicação mostra junto ao feedback qual porta USB está ocupada pelo *driver*. Se a conexão for mal sucedida, o feedback retorna um erro.

Depois de apresentar os dados recebidos do hardware Arduino, pode-se então escolher os botões para acionar alguma ação no hardware Arduino. Ao

clicar em SAIR, o sistema fecha a conexão com a porta Serial. A Figura 4-12 mostra o modelo da aplicação utilizada para testes no computador Desktop.

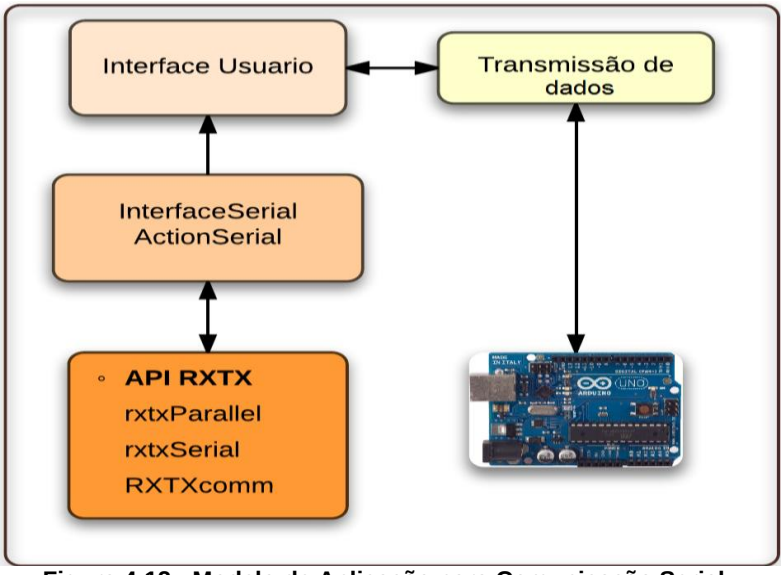


Figura 4-12 - Modelo de Aplicação para Comunicação Serial.

O sistema atendeu ao esperado tanto na comunicação via USB como com a utilização de radiofrequência. A tela utilizada como teste é apresentada na Figura 4-13.

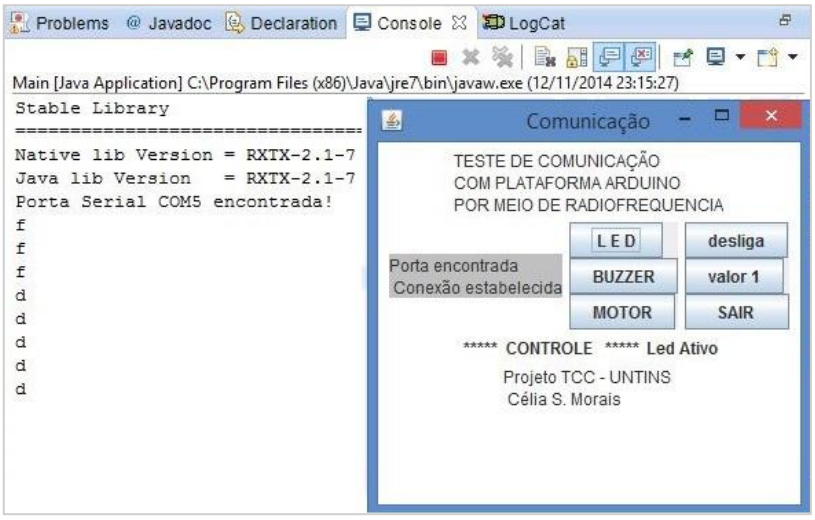


Figura 4-13 - Teste de comunicação com Arduino.

Tanto a comunicação Serial, como a comunicação com o módulo de rádio funcionaram perfeitamente com essa aplicação de teste. Para a alternância entre conexão USB e o módulo de rádio não houve necessidade de fazer qualquer tipo de alteração no código utilizado. O que houve de mudanças foram somente nas

configurações de hardware, que, como já fora explicado anteriormente, alguns valores exigem que sejam igualados em todos os meios que se comunicam, que nesse caso, são as duas placas de comunicação, e o software.

O sistema em Java para desktop funcionou como preparação para o trabalho direto no dispositivo móvel, que demandava estudos adicionais devido sua complexidade. Fato é que o computador possui uma interface serial nativa, o que facilita a comunicação USB. Já para utilização das interfaces de comunicação no sistema Android, são necessários *drivers* adicionais e interpretação da API com host USB, que funciona como controle de transferência em massa.

No entanto, o sistema Android envia uma string para o conversor, que, por sua vez, a converte em bytes que a placa Arduino reconheça. Essa ponte pode causar um *delay* maior que o alcançado na comunicação direta do desktop, e precisa ser tratado em código.

4.2.2 APLICATIVO ANDROID - JOYSDROID

A etapa de desenvolvimento do aplicativo para Android passou por dois momentos distintos. Um se refere ao desenvolvimento da lógica do sistema que permite a comunicação com o hardware externo e o outro diz respeito ao desenho da interface. Cada parte tem suas peculiaridades que merecem atenção diferenciada.

4.2.2.1 SISTEMA DE COMUNICAÇÃO COM HOST USB

O objetivo da aplicação é abrir a comunicação, verificar se houve resposta e retornar um feedback. Enquanto a conexão estiver ativa, o sistema possibilita o envio de dados por via USB para a porta serial do hardware. Se a conexão do hardware externo for interrompida, a conexão USB do Aplicativo é encerrada. O sistema pode também solicitar o fechamento da porta.

Para a comunicação USB no sistema Android, foi utilizado a biblioteca FTDriver e FTD_USB, ambas programadas com base na biblioteca nativa do Android: *Acessory USB*. A partir do nível API12 do Sistema Operacional Android é disponibilizada a biblioteca que trata todos os pontos importantes para conexão USB, o que auxilia na criação e utilização de *drivers* adicionais para

implementação da comunicação entre o dispositivo móvel e um periférico. O Quadro 9 traz as principais classes de comunicação nativas do Android que foram aplicadas a esse projeto.

Quadro 9 - Principais classes de comunicação USB Android.

QUADRO IX	
CLASSE	**DESCRIÇÃO**
UsbManager	Acessa o estado do USB e se comunica com os periféricos conectados.
UsbDevice	Faz a comunicação com o hardware periférico, desde que o dispositivo Android possua suporte a host USB.
UsbAcessory	Representa um acessório USB.
UsbDeviceConnection	Envia e recebe as mensagens de dados e de controle para o dispositivo USB.
UsbRequest	Representa uma solicitação

Fonte: ANDROID DEVELOPERS, 2014.

Durante o processo de desenvolvimento do aplicativo, outras bibliotecas foram testadas. Alguns resultados foram até satisfatórios enquanto os testes aconteciam com comunicação cabeada, que foi o caso da biblioteca *UsbSerialLibrary*. A biblioteca permite iniciar a conexão passando parâmetros como a porta a ser solicitada e quantidades de bits a ser enviado. A Figura 4-14 mostra parte do código responsável por abrir a conexão com o Arduino .

```
super.onResume();
Log.d(TAG, "Resumed, sDriver=" + sDriver);
if (sDriver == null) {
    mTitleTextView.setText("No serial device.");
} else {
    try {
        sDriver.open();
        sDriver.setParameters(115200, 8, UsbSerialDriver.STOPBITS_1,
                UsbSerialDriver.PARITY_NONE);
    } catch (IOException e) {
        Log.e(TAG, "Error setting up device: " + e.getMessage(), e);
        mTitleTextView.setText("Error opening device: " + e.getMessage());
        try {
            sDriver.close();
        } catch (IOException e2) {
            // Ignore.
        }
        sDriver = null;
        return;
    }
    mTitleTextView.setText("Serial device: " + sDriver.getClass().getSimpleName());
}
onDeviceStateChange();
```

Figura 4-14 - Exemplo de solicitação de conexão da Bilbioteca *UsbSerialLibrary.*

Outros sistemas de controle como o DROIPLANNER utilizam essa biblioteca como base para receber dados do VANT para a tela do dispositivo móvel. A biblioteca trata o atraso do recebimento de dados do buffer de maneira que tornam imperceptível o *delay* causado durante a transmissão. Nesse caso, o hardware precisa ser configurado somente para enviar dados para o aplicativo. O problema enfrentado no presente trabalho foi relacionado ao hardware e equipamentos disponíveis, já que a melhor forma de comunicação por essa biblioteca seria com módulos diferentes ao que estava disponível para esse projeto.

A biblioteca FTDriver, aplicada à implementação do aplicativo, foi desenvolvida por Keisuke Suzuki (2011). Foi projetada para controlar a comunicação serial pela ativação do chip FTDI, conversor serial USB, e possui suporte ao CDC-ACM, que é um protocolo que emula portas seriais por meio de USB, independente do fabricante. A parte do código que solicita a conexão é apresentada na Figura 4-15.

```
if(mySerial.begin(FTDriver.BAUD9600))
{
    conexão  = true;
    abrirConexao.setEnabled(false);
    encerrar.setEnabled(true);
    Toast.makeText(this, "CONEXÃO ESTABELECIDA", Toast.LENGTH_SHORT).show();
    help.setVisibility(View.INVISIBLE);
    conect.setVisibility(View.INVISIBLE);
    contTitle.setVisibility(View.VISIBLE);
}
```

Figura 4-15 - Solicitação de conexão USB.

O sistema desenvolvido permite o envio de dados por meio de conexão USB para o hardware que funciona como ponte entre o sistema Android e o hardware principal, que se comunicam por radiofrequência. A Figura 4-16 mostra o diagrama do funcionamento da comunicação.

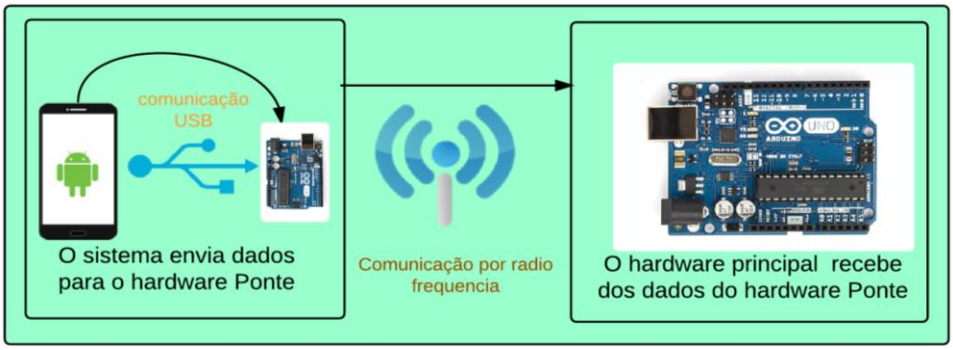


Figura 4-16 - Comunicação Android e Arduino.

Essa forma de comunicação permite o envio de dados com delays mínimos, que muitas vezes passam despercebidos. A problemática enfrentada aqui foi a questão de envio de dados do hardware principal para Android. Durante a implementação foram feitos testes para receber dados, já que a biblioteca utilizada permite o envio e recebimento de dados pela porta USB, mesmo que de forma assíncrona. O problema surgiu devido o uso de duas interfaces seriais de hardware - o hardware ponte e o hardware principal. Seria necessário estabelecer essa conversa entre os dois seriais com caminhos de ida e volta, mas as restrições da placa Arduino não permitiram, limitando o software a só transmitir dados, como definido no objetivo inicial.

4.2.2.2 A INTERFACE

A proposta da interface era possibilitar o controle de um VANT por meio de dois controles joysticks simulados na tela do *tablet*. Alem de atender aos requisitos do sistema, foi imprescindível pensar sob a ótica do usuário. Questões como a distancia entre os controles para melhor usabilidade das mãos e quantidade de informações necessárias na tela foram estudadas e trabalhadas durante a execução do projeto.

O sistema possui tela inicial com menu de navegação que contem: Ajuda, Iniciar, Informações e opção de Sair, conforme mostra a captura de tela na Figura 4-17.

Figura 4-17 - Tela de Menu Inicial do Joysdroid.

O fluxo de telas definido para o aplicativo é apresentado na Figura 4-18.

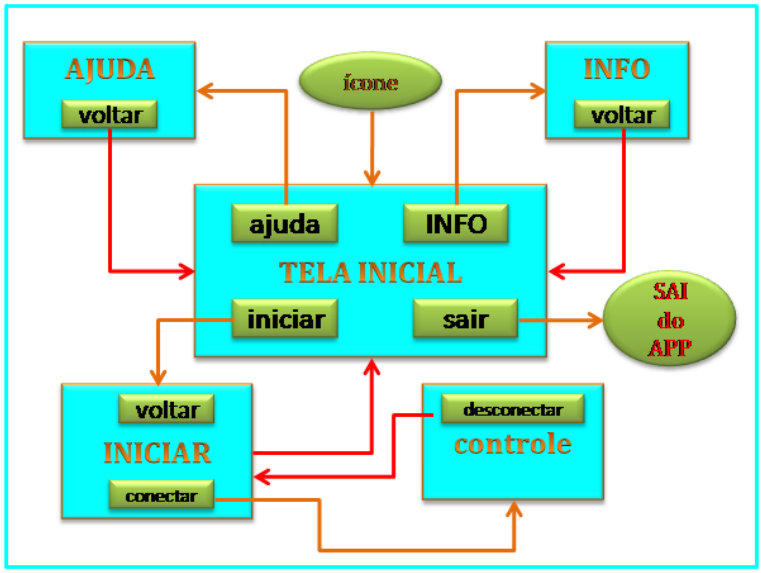


Figura 4-18 - Fluxo de telas do aplicativo Joysdroid.

A tela de ajuda apresenta informações pertinentes ao uso do aplicativo, detalhando a forma para estabelecer uma conexão com hardware externo. Ao conectar o cabo USB no dispositivo, é necessário solicitar permissão para estabelecer a conexão. Depois dessa confirmação é que se pode iniciar com o envio de dados pelo controle.

A tela principal, que se refere ao controle joystick, foi desenvolvida com base no projeto *mobile-anarchy-widgets.* A interface simula um joystick na tela e retorna as coordenadas de posição conforme seu movimento. Para o desenho do controle joystick, é utilizado as classes nativas do Java: Paint e Canvas. A Figura 4-19 mostra o método que realiza o desenho do controle joystick.

```
@Override
protected void onDraw(Canvas canvas) {
        int px = getMeasuredWidth() / 2;
        int py = getMeasuredHeight() / 2;
        int radius = Math.min(px, py);
        // Desenhe o fundo
        canvas.drawCircle(px, py, radius - innerPadding, circlePaint);

        // Desenhe a alça
        canvas.drawCircle((int) touchX + px, (int) touchY + py, handleRadius,
                        handlePaint);
        canvas.save();
}
```

Figura 4-19 - Método que Faz o Desenho do controle joystick.

Além do desenho, o projeto define o movimento do circulo menor e garante sempre seu retorno ao centro, como deve ser um controle joystick. Para aplicar ao projeto, foi necessário implementar funções para os dois controles joysticks, pois cada movimento gerado representa um byte para ser enviado ao Arduino. Como o retorno da interface do controle são pontos de coordenadas, foi definido um protocolo para receber apenas quatro posições de cada controle com base na posição do desenho do joystick na tela. Os valores das coordenadas são definidos em um plano cartesiano, de forma a fazer um giro de 360º.

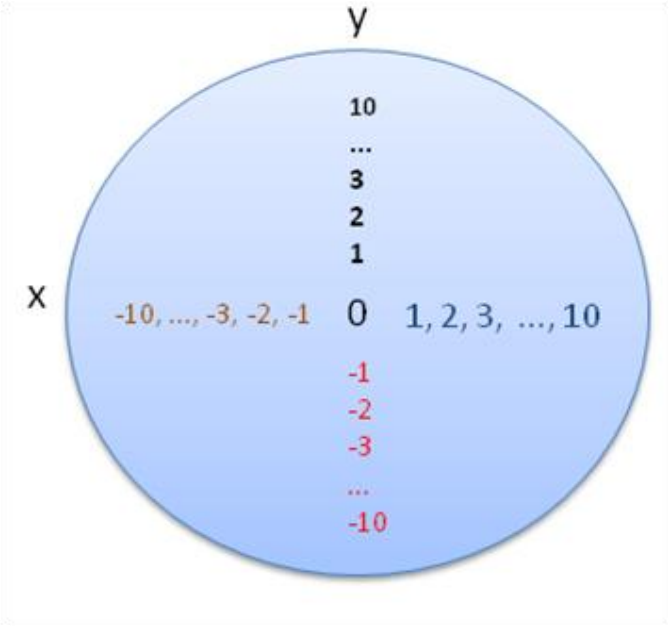


Figura 4-20 - Coordenadas do Joystick

O Quadro 10 mostra a definição para as posições do joystick.

Quadro 10 - Posições definidas para o controle.

QUADRO X			
POSIÇÃO	**INTERVALO DE PONTOS**	**INDICAÇÃO**	**SIMBOLO**
1	Y = -10 ^ (-10 < X < 10)	Pra cima	+
2	Y = 10 ^ (-10 < X < 10)	Pra baixo	-
3	X = 10 ^ (-10 < Y < 10)	Esquerda	>>
4	X = -10 ^ (-10 < Y < 10)	Direita	<<

O sistema de construção do VANT do tipo quadrotor traz definido as variáveis para os 8 movimentos. Trata-se de um *array* de 0 a 7, em que cada posição é responsável por uma série de eventos que vai desde o acionamento do motor até a posição do movimento da hélice. Como esse trabalho traz somente valores simulados, as variáveis tratadas aqui são responsáveis pelo acionamento dos componentes. Para tanto, foi definido ações para cada posição do controle joystick com base no protótipo de hardware citado nas seções anteriores. O protótipo disponibiliza os seguintes componentes :

- 1 LED
- 1 BUZZER
- 1 Servo Motor.

São oito comandos transmitidos para o hardware que tem suas ações definidas conforme o Quadro 11.

Quadro 11 -Movimentos e Ações do Controle.

QUADRO XI				
controle	**posição**	**Byte enviado**	**valor**	**ação**
Controle Joystick Direito	1	f	1	LED HIGH
	2	g	2	LED LOW
	3	d	3	BUZZ AGUDO
	4	e	4	BUZZ MÉDIO
Controle Joystick Esquerdo	1	a	5	GIRO RÁPIDO
	2	c	6	GIRO LENTO
	3	b	7	GIRO ESQUERDA
	4	a	5	GIRO DIREITA

Para o controle de um VANT do tipo quadrotor, como definido no objetivo do projeto, são necessários apenas 8 movimentos essenciais. A Figura 4-21 mostra um modelo de como pode ser utilizado os controles joysticks do aplicativo para o VANT, de forma que atenda a todos os movimentos necessários.

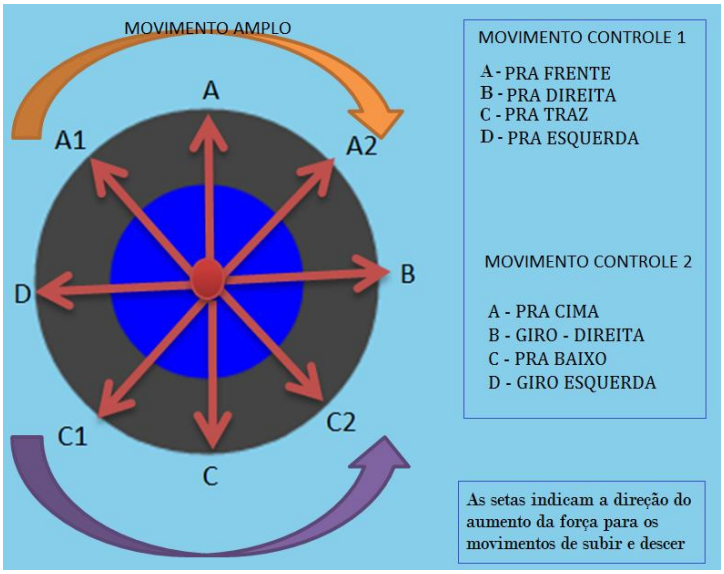


Figura 4-21 - Modelo para configuração do Controle Joystick.

A tela principal do aplicativo é apresentada na Figura 4-22.

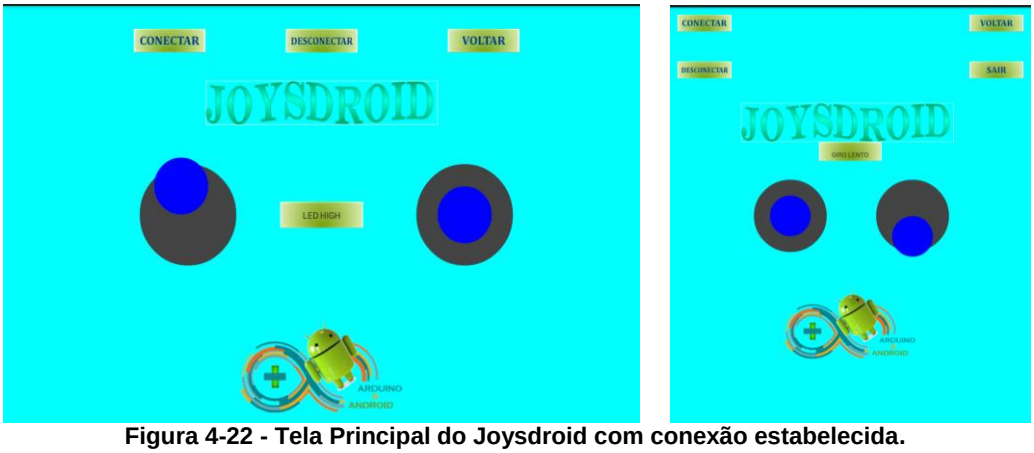


Figura 4-22 - Tela Principal do Joysdroid com conexão estabelecida.

O movimento do joystick é possibilitado mesmo sem a conexão estabilizada. Nesse caso, cada movimento representa um símbolo conforme apresentado no Quadro 10. Quando a tela está aguardando conexão, uma imagem de instrução para conectar ao comunicador fica visível.

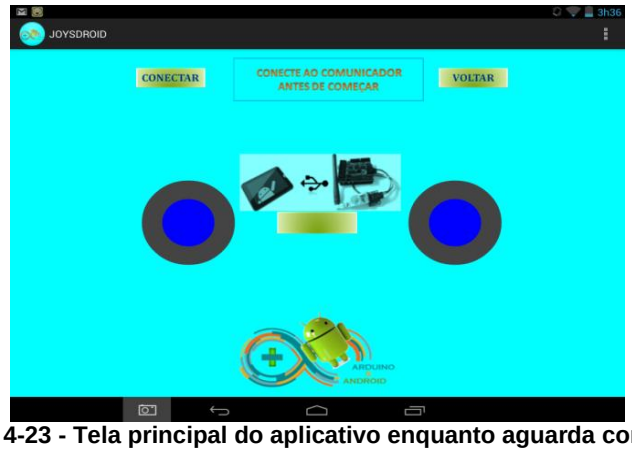


Figura 4-23 - Tela principal do aplicativo enquanto aguarda conexão.

A instrução apresentada na tela principal é para deixar claro que o sistema só vai receber dados para envio se houver uma conexão estabelecida. Os controles joysticks funcionam normalmente sem a conexão, mas sem armazenar valores. Os dados que aparecem quando acionados nesse caso, se referem a dados estáticos que mostram a direção do movimento, conforme apresentado na coluna "símbolos" do Quadro 10 nessa seção.

4.2.2.3 CONFIGURAÇÕES E RESTRIÇÕES DO SISTEMA

Os parâmetros adotados para a simulação são: *baud rate* 9600, necessário para conectar à porta, que deve ser igual nas configurações do módulo de rádio e no Sketch. As variáveis padronizadas são: 'a', 'b', 'c', 'd', 'e', 'f' e 'g', conforme Quadro 11, que devem estar presentes no Sketch para funcionamento do aplicativo.

Figura 4-24 - Conexão com cabo USB OTG.

É necessário que o dispositivo móvel tenha suporte a conexão USB-OTG, para habilitar o *host-USB*, função disponível em alguns aparelhos com Android 3.2 ou superior. O aplicativo foi desenvolvido para dispositivos com telas de 7" ou superior.

4.3 TESTES E ANÁLISE DOS RESULTADOS

Como já citado, o presente trabalho teve início com estudos voltados para conhecimento dos VANTs do tipo quadrotor, funcionamento da aerodinâmica dessas aeronaves, curiosidades de sua utilização e sistema de controle. Esse aprendizado foi imprescindível para direcionar o objetivo central do projeto que era a criação de um sistema de controle remoto para VANT.

Inicialmente os experimentos se limitavam a abrir a conexão da porta Serial, reconhecer o dispositivo, enviar os bits de dados e receber uma resposta. As etapas de testes foram:

1. Arduino X Computador
    a. IDE Arduino - USB
    b. IDE Arduino - RF
    c. Aplicação Java - USB
    d. Aplicação Java - RF
2. Arduino X Arduino
    a. IDE Arduino - USB
    b. IDE Arduino - RF
3. Arduino X Android USB
    a. USB
    b. RF

As duas primeiras etapas de testes sem dispositivo móvel foram importantes para definir os parâmetros necessários para a comunicação direta via USB, e posteriormente, via radiofrequência. Pontos importantes como tempo de atraso na transmissão diferenciado entre as duas formas de conexão puderam ser verificados e corrigidos de forma a atender os requisitos do sistema.

O resultado do experimento de transmissão de dados com a melhor performance encontrada é apresentado no Quadro 12.

Quadro 12 - Experimento de Transmissão de dados.

QUADRO XII				
Arduino x		**Delay configurado**	**Delay real**	**Atraso de transmissão**
IDE Arduino	**USB**	0,5"	0,5"	0
	RF	0,5"	0,5"	0
Java Desktop	**USB**	0,5"	0,5"	0
	RF	2"	3"	1"
Arduino	**IDE**	0,5"	0,5"	0
	RF	0,5"	0,5"	0
Android	**USB**	1"	3"	2"
	RF	1"	2"	1"

Nessa etapa, a perda de dados foi notória somente para a transmissão entre o aplicativo android e o protótipo de hardware via radiofrequência. Em uma simulação de 5 minutos, foi observado que pelo menos 15 comandos não chegaram ao destino. Apesar de parecer uma quantidade grande de dados perdidos, esse valor representa em média de 5%, quando se leva em consideração que o controle, em seu menor desempenho, é capaz de enviar pelo menos um comando por segundo. O fato é que se cogitar o uso desses dados com relação ao controle de um VANT seria de grande importância que não tivesse essa quantidade de dados perdidos, visto o nível de perícia que se refere ao controle de uma aeronave. A Figura 4-25 mostra um momento de Simulação.

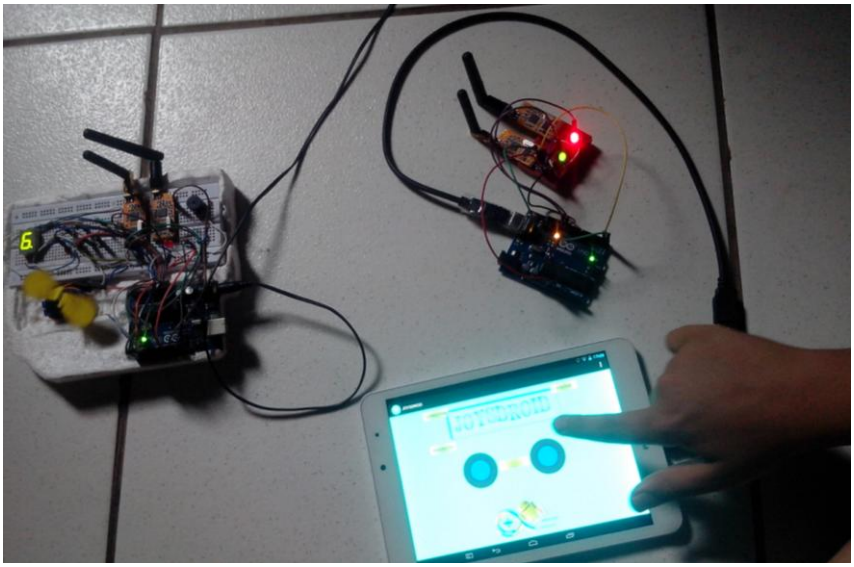


Figura 4-25 -Simulação do Controle.

Foram realizados teste de transmissão e recebimentos de dados para o sistema, porém o resultado não foi satisfatório, conforme pode ser observado no Quadro 13. A ideia era receber informações sem interferir nos dados enviados, de forma síncrona.

Quadro 13 - Experimento de transmissão e recepção de dados.

QUADRO XIII					
Arduino x		**Delay configurado**	**Delay real**	**Atraso de Envio**	**Atraso de Recebimento**
IDE Arduino	**USB**	0,5''	0,5''	0	0
	RF	0,5''	0,5''	0	0
Java Desktop	**USB**	0,5''	1'' a 2''	0	1''
	RF	2''	2'' a 5''	1'' a 2''	2'' a 4''
Arduino	**IDE**	0,5''	0,5''	0	0
	RF	0,5''	0,5''	0	0
Android	**USB**	5''	5'' a 7''	1'' a 3''	4'' a 7''
	RF	10''	15''	2'' a 3''	10'' a 15''

Para simular o recebimento de dados, acrescentou-se um sensor de temperatura ao protótipo com o objetivo de enviar o valor da temperatura atual com o acionamento de um botão. O protótipo é apresentado na Figura 4-26.

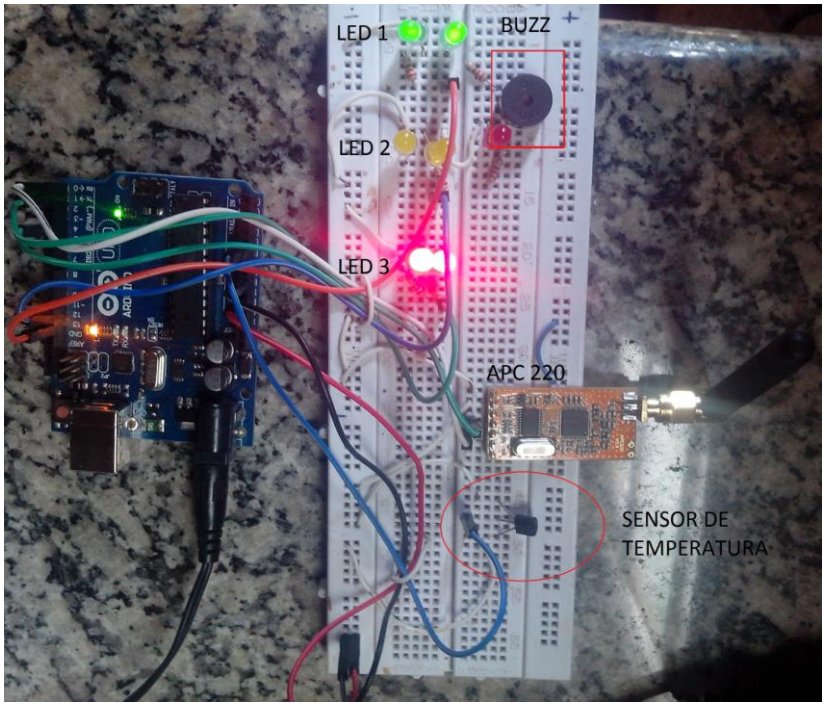


Figura 4-26 - Protótipo de hardware para envio e recebimento de dados.

Para os testes realizados no computador a simulação funcionou sem problemas de atraso, porém, no dispositivo móvel o tempo para envio do comando somado ao tempo de espera pela resposta chegou a até 15 segundos. Um outro experimento foi realizado para verificar a perda de dados, mas nesse caso, o sensor de temperatura enviava os valores automaticamente ao aplicativo a cada 10 segundos, e nesse período, comandos eram enviados do aplicativo para o protótipo de hardware para ativar algum componente. O esquema do protótipo de hardware para esse experimento em particular é apresentado na Figura 4-27.

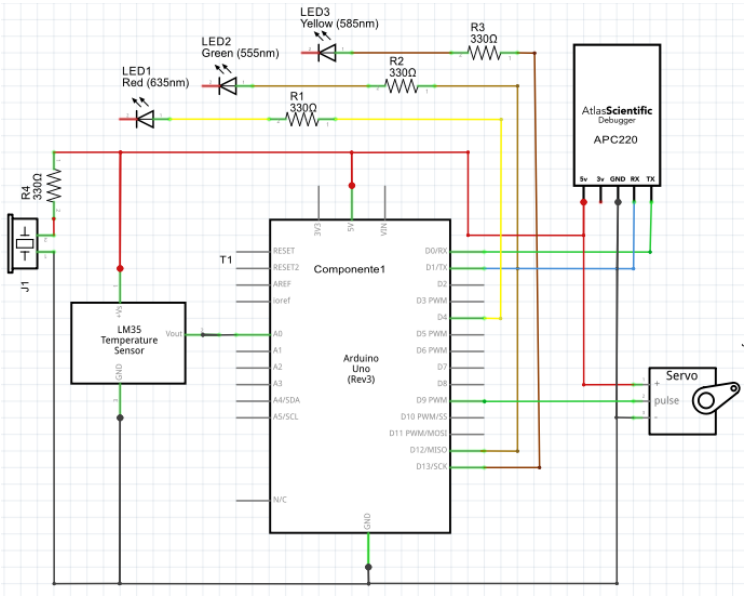


Figura 4-27 - Esquema do protótipo de hardware com sensor de temperatura.

Os valores recebidos pelo dispositivo de hardware no sistema de controle se referia a um vetor de 5 posições em que cada uma correspondia a uma parte do valor da temperatura: 32.75, por exemplo.O resultado desse teste foi que a cada 3 entradas de dados no aplicativo Android, apenas uma informação chegava completa, pois parte dos valores se perdiam com a repetição de envio. Com os comandos enviados ao hardware a perda de dados também foi significativa. Em alguns momentos era necessário repetir diversas vezes o mesmo movimento no controle para a ação acontecer na plataforma de hardware. Esse experimento em particular, só foi possível com a utilização da biblioteca *usbSeriallibrary* para android, conforme explicado na seção de desenvolvimento do software.

Quando se tratou de estabelecer uma conversa entre Arduino e outros

ambientes por meio de cabo USB, a comunicação é transparente e praticamente não há perda de dados, mesmo quando o trabalho com a comunicação é paralela. O problema surge ao estabelecer a conversa com um dispositivo móvel por meio de módulo sem fio, em particular com o módulo APC220, que foi o único meio utilizado nesse projeto. A solução para esse problema seria utilizar uma Shield com módulo para Bluetooth, para o caso de conexões de curta distancia, ou Wi-Fi, para conexões com distancias maiores, como no caso do VANT, que permitem, por exemplo, que um módulo seja configurado para receber os dados enquanto outro módulo ficaria reservado somente para a transmissão.

Com a analise dos resultados desses testes foi definido finalizar o projeto somente com transmissão de dados para o protótipo de hardware, alcançando o objetivo principal do projeto. No entanto, quando se trata de controlar a navegação de uma aeronave independente de sua categoria, qualquer atraso no envio e recebimento de dados pode causar inúmeros problemas.

Nota-se que mesmo com o trabalho só de transmissão de dados houve atraso de até 2 segundos (Ver Quadro 12). Parte dsse atraso na transmissão de dados do Aplicativo Android para a plataforma Arduino foi resolvido com a utilização de um protótipo que funciona como ponte para a comunicação, como explicado no capítulo 4, seção 4.1.

Dessa forma, o tempo entre o inicio da transmissão pelo acionamento do controle no dispositivo móvel até o inicio da ação na plataforma de hardware pode chegar a até 0,5 segundos, o que é imperceptível para testes locais e sem movimento. O esquema do protótipo principal finalizado é apresentado na Figura 4-28.

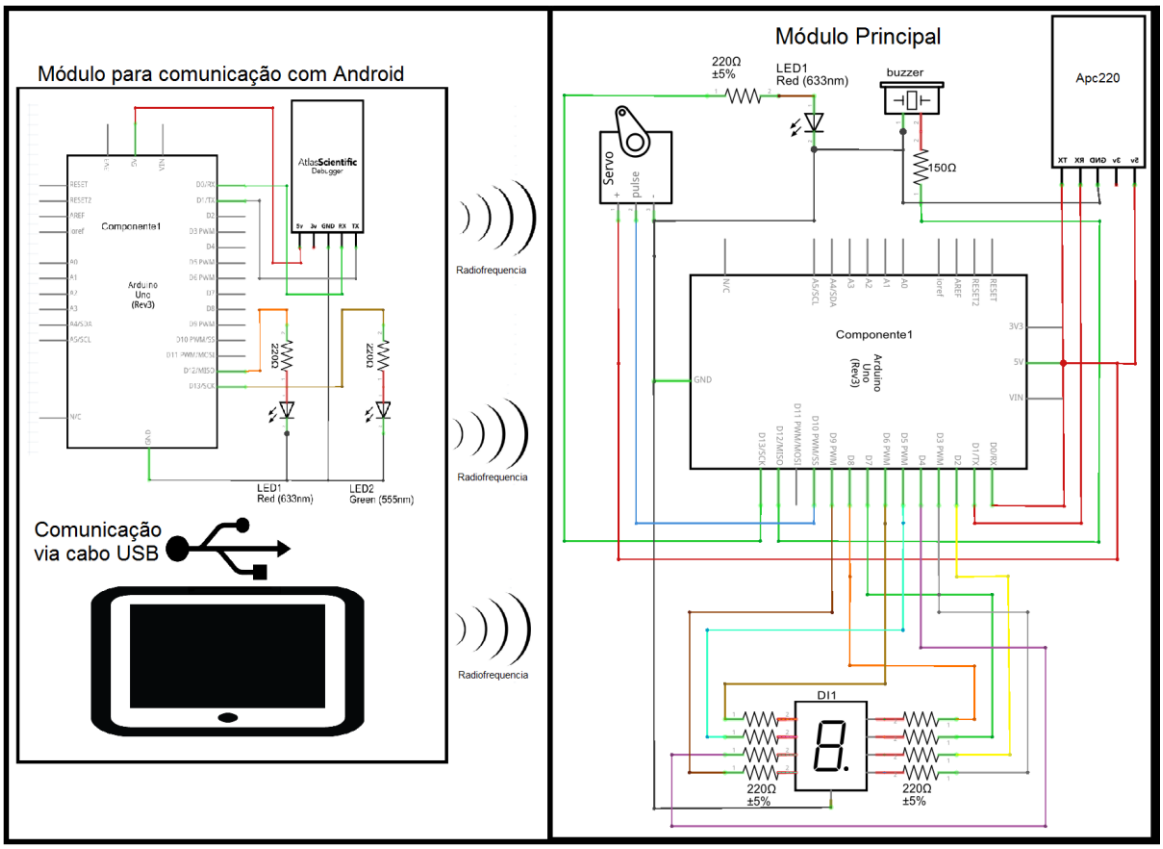


Figura 4-28 - Esquema do Protótipo final de Hardware.

Um dos problemas enfrentados durante o desenvolvimento do projeto foi a falta de equipamentos indispensáveis para os experimentos. Módulos de Bluetooth ou Wi-Fi para o hardware Arduino seriam essenciais para estabelecer uma comunicação eficiente entre o Android e protótipo de hardware. Outro item importante é a placa CRIUS AIO (All in One) 1.1, que possibilita configurar um controlador de voo. A falta desse componente em especial, inviabilizou experimentos de simulação que teriam as variáveis reais de um VANT. Por esse motivo, os resultados apresentados são viáveis para experimentos quando se trata de estabelecer uma conversa entre o dispositivo móvel e a plataforma Arduino com outras finalidades, porém, para controlar a navegação de um VANT é necessário a inserção de componentes e equipamentos adequados para que garanta o resultado preciso e seguro no que tange ao controle de uma aeronave.

5. CONCLUSÕES

O desenvolvimento desse trabalho envolveu praticas de programação para o desenvolvimento do aplicativo em Android e conceitos de robótica para a construção de protótipos na plataforma Arduino. A integração das tecnologias envolvidas foi um desafio um tanto interessante, uma vez que os componentes disponíveis limitavam as alternativas para atingir o objetivo do projeto.

Em função do objetivo, foram construídos protótipos de hardware na plataforma Arduino com a finalidade de testar as funcionalidades do aplicativo, bem como a eficiência do sistema. A problemática envolvida nessa etapa era a busca por soluções que simplificassem a comunicação do protótipo com um ambiente exterior ao IDE Arduino, por meio de radiofrequência. Como citado no trabalho, a falta de componentes restringiu a execução de alguns experimentos essenciais ao aperfeiçoamento do sistema, porém não impediu que os testes fossem realizados e o trabalho finalizado dentro do que fora definido no objetivo do projeto.

Todo o processo de implementação do aplicativo em Android foi realizado com base nos estudos feitos em sala de aula durante o curso, o qual disponibilizou duas disciplinas dedicadas ao desenvolvimento em Android nos períodos de 2013/2 e 2014/1, respectivamente. Os conceitos passados nas aulas aliados à prática aplicada foram suficientes para sanar as dúvidas que apareceram no decorrer do projeto.

O aplicativo desenvolvido possibilita ao usuário iniciar uma conexão com modo host USB e enviar comandos para uma plataforma de hardware através de um comunicador, por meio de radiofrequência. Os testes aplicados junto à plataforma de hardware foram satisfatórios, pois atendem a necessidade de enviar comandos com o menor tempo possível, evitando assim problemas com perda de dados. No entanto, no que se refere ao controle real de uma aeronave, ficou uma grande lacuna devido às limitações encontradas com a utilização dos componentes de hardware disponíveis para testes. Porém, se pensar do lado da solução com componentes básicos e mais econômicos para comunicação entre a plataforma Arduino e o sistema Android, o trabalho traz resultados aceitáveis que

provam sua eficácia para diversos outros tipos de aplicações.

Conhecer a história do VANT e como a evolução da tecnologia implica diretamente na popularização dessas aeronaves fez com que a pesquisa tomasse um rumo mais dinâmico e colaborativo, pois envolveu contato direto com outros pesquisadores que trabalham ou já trabalharam com construção de Drones, bem como outros que atuaram em áreas relacionadas ao tema. O envolvimento com o projeto proporcionou a oportunidade de trocar experiências com alunos de instituições externas a do projeto e ainda sanar dúvidas diversas que apareceram durante todo o processo de pesquisa e desenvolvimento do software.

Como trabalhos futuros, espera-se realizar a comunicação com a utilização de componentes adequados, o que vai possibilitar experimentos com VANTs já construídos. Outro ponto a ser estudado está relacionado ao envio de informações para ser embarcado no mesmo projeto. As plataformas populares para desenvolvimento de VANT trazem pré-programados o envio de dados por meio de radiofrequência, o que possibilita o recebimento de informações importantes sobre a localização, por exemplo, na aplicação. A problemática está em fazer o caminho inverso ao mesmo tempo.

Aliado ao desenvolvimento desse projeto, despertou-se o interesse pelo campo de pesquisa, o qual envolve busca de informação e interpretação de dados pertinentes à área estudada. Com a evolução tecnológica dos dias atuais, abre-se uma vasta oportunidade de aplicação do conhecimento já adquirido, e o mais importante, favorece a busca por novas experiências e competências.

Espera-se que a trajetória experimentada para desenvolver esse trabalho possa ser aproveitada para experiências futuras em que se sujeita a inserção de inovações na integração da tecnologia móvel e aplicações para VANTs.

6. REFERÊNCIAS

AMERICAN GENESIS - **"Um século de invenção e entusiasmo Tecnológico".** Disponível em: < http://www.studyblue.com/notes/note/n/us-history-test-1-pictures/deck/743090> Acesso em: 22 de maio 2014.

ANAC - Agencia Nacional de Aviação Civil - **2º workshop de regulamentação de VANTs.** Disponível em: <http://www2.anac.gov.br/arquivos/pdf/apresentacao2Workshop.zip> Acesso em: 29 de Abril de 2014.

ANDERSON, A.; AUGUSTO, F**. Introdução ao Desenvolvimento Mobile com Android**. (2012). ANAIS - V Época - Natal/RN, Brasil. Disponível em: <http://www.lasic.ufrn.br/epoca2012/tiki-download_file.php?fileId=13> Acesso em: 10 de julho de 2014.

ANDROID DEVELOPERS. **The developer's guide**. Disponível em: <http://developer.android.com/guide/index.html>. Acesso em: 24 de janeiro de 2014.

AUSTRALTECH, Ingenieria y Servicios EIRL. **Domótica Smarthouse**. Disponível em: < http://www.australtech.cl/servicios.html > Acesso em: 16 de novembro de 2014.

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

BENEMANN, A. **Estação De Controle Para Veículos Aéreos Não Tripulados** – Universidade do Rio Grande do Sul – Escola de Engenharia – Departamento de Engenharia – Porto alegra, 2013.

BRASSCOM - Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação **- Mobilidade, Inteligência de Mercado**. Periódico. Disponível em:< http://www.brasscom.org.br/brasscom/Portugues/download.php?cod=633> Acesso em: 20 de julho de 2014.

BRESSAN, M. B. **Introdução à Plataforma Arduino -** Minicurso. IFTO - Intituto Federal de Educação, Ciencias e Tecnologia. Sudeste de Minas Gerais. Campus de Juiz de Fora - MG, 2014.

_____Diário Oficial da União número 112 – seção 1. Portaria Normativa n.606/MD, de 11 de junho de 2004. Brasília, 14 jun 2004.

DX - Deal Extreme. Disponível em: < http://www.dx.com/pt/> Acesso em: 10 de julho de 2014.

FBS ELETRONICA. **Apostila Arduino - Com aplicações baseada na placa Arduino UNO**. Disponível em: < http://s3.amazonaws.com/ppt-download/apostilaarduinov0rv1fbseletronica-131023071049-phpapp01.pdf > Acesso em: 21 de setembro de 2014.

FENAPEF - Federação Nacional dos Policais Federais. - Disponível em: < http://fenapef.org.br/fenapef > Acesso em: 03 de outubro de 2014.

FERREIRA, J. M. **Introdução ao Projeto com Sistemas Digitais e Microcontrolados** (1ª ed.). Faculdade de Engenharia. Universidade do Porto. Portugal: 1998.

FURTADO, HUGO V. et al. **Aspectos de Segurança na Integração de Veículos Aéreos não Tripulados (VANT) no Espaço Aéreo Brasileiro**. Simpósio de Pesquisa em Transporte Aéreo, 2008, Rio de Janeiro. Anais do VII SITRAER. Rio de Janeiro: E-Papers, 2008.

GUIMARÃES, J.P.F., **Controle de Atitude e Altitude Para Um Veículo Aéreo Não Tripulado Do Tipo Quadrotor**. Universidade Federal do Rio Grande do Norte. Programa de Pós Graduação em engenharia elétrica e de computação. 2012.

HARDGRAVE, **O pioneirismo com o VANT;** Disponível em: <http://www.ctie.monash.edu.au/hardgrave/>. Acesso em 22 março de 2014.

IDC Brasil. **Analise de Mercado -** Mobile Devices/Tablets. Disponível em: <http://br.idclatin.com/> Acesso em: 07 de outubro de 2014.

LEISHMAN, J. G**.** ***Principles of helicopter aerodynamics***. 2 edn. Cambridge University Press. 2006. Disponível em: <http://www.air.flyingway.com/faaexam/heli/**principles_of_helicopter_aerodynamics**.pdf>

LEMOS, M. **Arduino: Conheça esta Plataforma de Hardware Livre e suas Aplicações**. Artigo. Disponível em: < http://blog.fazedores.com/arduino-conheca-esta-plataforma-de-hardware-livre-e-suas-aplicacoes/> Acesso em: 10 de agosto de 2014.

MANUAL APC220. **APC220 Radio Data Module**. Versão 1.3. 2010. Disponível em: < http://www.dfrobot.com/image/data/TEL0005/APC220_Manual_en.pdf > Acesso em: 10 de agosto de 2014.

MICROCHIP-MCP73831 - Disponível em: <http://www.microchip.com/wwwproducts/ Devices.aspx?dDocName=en024903> acesso em: 23 de junho de 2013.

MOBILE ANARCHY. **Mobile-Anarchy-Widgets**. Projeto. Disponível em: < https://code.google.com/p/mobile-anarchy-widgets/wiki/JoystickView>. Acesso em: 21 de maio de 2013.

MONTEIRO, J. B. **Google Android - Crie Aplicações para Celulares e Tablets.** Casa do Código, 2012. São Paulo - SP.

NERY,T. **Estudando Android - Activity e seu Ciclo de Vida**. (2012) Artigo. Disponível em: < http://javarockexception.wordpress.com/2012/03/12/estudando-android-ciclo-de-vida-activity/ > Acesso em: 10 de julho de 2014.

NONAMI, K., F. KENDOUL, S. SUZUKI,W.WANG & D. NAKAZAWA (2010), **Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles, Springer**.<http://books.google.com.br/books?id=6H-ANYxUKzwC>

OLIVEIRA, A, P. **Neurais Artificiais**. Dissertação de Mestrado. Centro de Tecnologia. Universidade Federal do Rio Grande do Norte. Brasil: 2012.

OLIVEIRA, F, A,. **CTA e o Projeto VANT**. In: 1° Seminário Internacional de Vant. São José dos Campos, 2005. Palestra proferida no Centro Tecnológico da Aeronáutica em 11 jun 2005.

OTOBONNI, J. **VANTS - Brasil Está Na Vanguarda No Uso Pela Polícia Em Termos Mundiais** (2014).
Disponível em: < http://www.defesanet.com.br/vant/ > Acesso em: 28 de maio de 2014.

PAULA V. M. G. **Bqm-1br: O VANT _a jato brasileiro**. disponível em: <http://www.ecsbdefesa.com.br/defesa/fts/BQM1BR.pdf.> Acesso dia 10 de janeiro de 2014.

PROMON. **Mobilidade - A grande Tendência do Futuro** - Periódico: *Busines &Technology Review*. Disponível em: <http://www.teleco.com.br/promon/pbtr/Mobilidade_4Web.pdf> acesso em: 20 de Julho de 2014.

SÁ, R. C. **Construção, modelagem dinâmica e controle Pid para estabilidade de um veículo aéreo não tripulado do tipo quadrotor**. Universidade Federal do Ceará, Fortaleza, 2012.

SENAI - Serviço Nacional de Aprendizagem Industrial**. Fundamentos de Rádio Frequência**. Curso de Redes sem Fio. Disponível em: <http://pt.slideshare.net/carlosvmelo/fundamentos-de-radio-freqncia> acesso em: 10 de janeiro de 2014.

SILVA, F. **Comunicação Serial - Arduino e Game Maker** - Tutorial. Disponível em: <https://www.ufmg.br/frmfa/wpcontent/uploads/2012/07/TutorialComunica%C3%A7%C3%A3o-Arduino-e-Game-Maker1.pdf > Acesso em: 03 de outubro de 2013.

SILVA, P. R. V. R. Utilizando a API RXTX para manipulação serial. Disponível em: < http://www.devmedia.com.br/utilizando-a-api-rxtx-para-manipulacao-da-serial-parte-i/6722> Acesso em: 03 de maio de 2014.

SOUSA, J. D. A. ***Development of Unmanned Aerial Four-Rotor Vehicle***. Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores Major Automação. Faculdade de Engenharia da Universidade do Porto, Portugal: 2011.

SPINOLA, E. O. **Android, a nova plataforma móvel - Parte IV.** (2008). Artigo. disponível em: < http://www.devmedia.com.br/android-a-nova-plataforma-movel-parte-iv/8434 > Acesso em: 10 de julho de 2014.

SUZUKI, K. KsKsue - Projeto GitHub. **FTDriver**. Disponível em: < https://github.com/ksksue/FTDriver > Acesso em: 10 de agosto de 2014.

TANENBAUM A.S; WETHERALL D. **Redes de Computadores**", São Paulo: Pearson Prentice Hall, 2011.

VASCONCELOS, C,S. **Projeto, Construção e Controle de quadrotor**. UFRJ - Departamento de Eletrônica e Computação, 2013.

XCUBE LABS. **The Android Story**. Disponível em: < http://www.xcubelabs.com/the-android-story.php > Acesso em: 20 de setembro de 2014.

ANEXOS

ANEXO 1 - Physicaloid Biblioteca

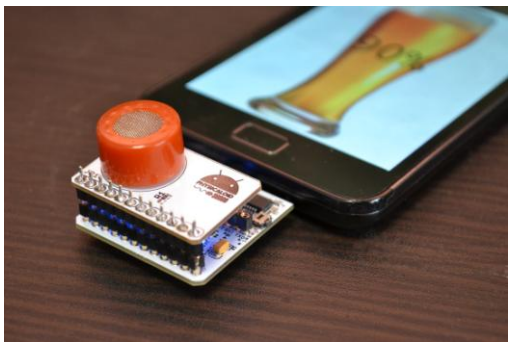
Trata-se de uma biblioteca desenvolvida com a função de se comunicar direto com placas físico-computacionais, como o Arduino. De acordo com o desenvolvedor, é possível, com essa biblioteca, incluir firmeware Arduino direto no aplicativo Android, sem passar pelo computador e fazer upload para o Google Play.

Suas características são:

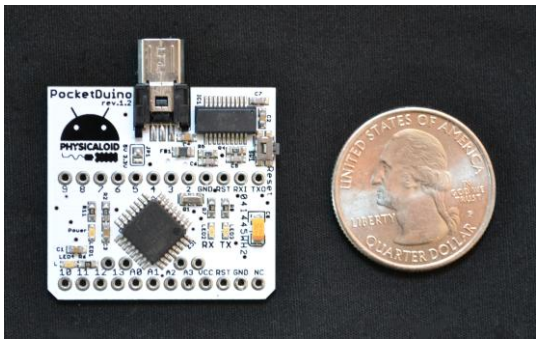
- Projeto de biblioteca Java Android
- A comunicação serial USB
- Pode carregar um firmware para um Arduino
- Suporte a Android 3.1 ou superior (USB precisa recurso Host API)
- não requer ROOT
- Suporte a CDC-ACM, FTDI, Silicon Labs CP210x
- Suporta upload de protocolos stk500, STK500V2
- Open-source (Apache License 2.0)

Mesmo com toda tecnologia empregada nesse projeto, ainda há a necessidade do contato direto da placa Arduino por meio da conexão USB com o

Android, para que o aplicativo funcione, como pode ser visto, com seu uso no projeto PocketDuino.



O PocketDuino se refere a um projeto com uma placa pequena que tem o objetivo de incrementar a internet das coisas. A pequena placa funciona da mesma forma que um Arduino, com a diferença que é prática e cabe no bolso, podendo assim ser carregada para qualquer lugar. A placa já tem a entrada para USB compatível com USB micro-b



Suas características são:

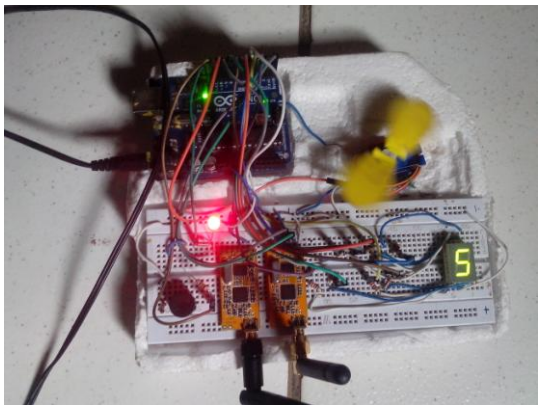
- Arduino Pro Mini pin-compatível
- Tamanho - 1,2" por 1,2" (3 cm)
- Não precisa de bateria extra para o Android
- Open-Source - código disponível no GitHub
- Utiliza biblioteca Java: PhysicaloidLibrary
- Baixa o firmware direto na placa
- Não tem suporte a iPhone/iPad.

Alguns projetos com PocketDuino utilizando a biblioteca Physicaloid estão disponíveis em vídeo no site do PocketDuino.

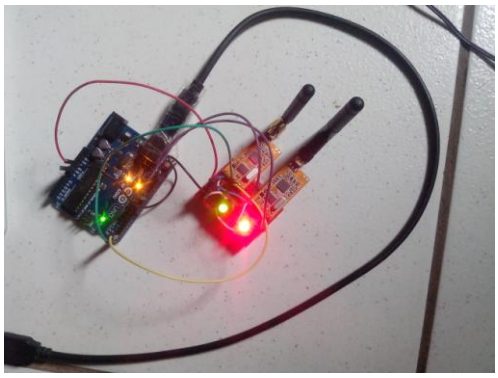
APENDICES

APÊNDICE A - Testes Para Comunicação Paralela com 2 Módulos de Rádio

A busca por um meio de sincronizar a comunicação entre o dispositivo móvel e a plataforma Arduino, levou a experimentos variados, incluindo a aplicação com 2 pares de APC220. Para a comunicação funcionar, a placa RF 1 - Referente a RX, estará configurada no hardware principal em TX, e no secundário em RX. Isso garante que o hardware secundário só envie para o principal. A placa RF 2 - Referente ao TX é configurada no hardware principal em RX, para garantir o recebimento de dados.



Protótipo do Hardware Principal configurado com 2 módulos de radio



Hardware secundário configura com 2 módulos de radio

Essa parte do trabalho ainda está em estudo, para que obtenha a melhor configuração com tempo de resposta dentro das possibilidades de uso do controle.

APÊNDICE B - Sketch dos Protótipos

PROTÓTIPO PRINCIPAL

```
/**************************************
autora: Célia da Silva Morais
Programa para ambiente de teste
**************************************/
/**************************************
São 5 componentes principais:
Servo motor = ligado ao pino 10
Buzzer = ligado ao pino 12
Led = ligado ao pino 13
APC220 = liagado RX para TX e TX para RX
Display de 7 seguimentos = pinos 2 ao 9
**************************************/
#include <Servo.h>
int pMotor = 10;
int pBuzzer = 12;
int pLed = 13;
Servo motor;
char x;
//Cria matriz para acionar acionar os segmentos
byte matrizDisplay[10][7] = {
  {1, 1, 1, 1, 1, 1, 0}, // = Digito 0
  {0, 1, 1, 0, 0, 0, 0}, // = Digito 1
  {1, 1, 0, 1, 1, 0, 1}, // = Digito 2
  {1, 1, 1, 1, 0, 0, 1}, // = Digito 3
  { 0, 1, 1, 0, 0, 1, 1}, // = Digito 4
  {1, 0, 1, 1, 0, 1, 1},  // = Digito 5
  {1, 0, 1, 1, 1, 1, 1},  // = Digito 6
  {1, 1, 1, 0, 0, 0, 0},  // = Digito 7
  {1, 1, 1, 1, 1, 1, 1},  // = Digito 8
  {1, 1, 1, 0, 0, 1, 1},  // = Digito 9
};
void setup(){
 Serial.begin(9600);
  pinMode(pBuzzer, OUTPUT);//pino 11 para o Buzzer
  pinMode(pLed, OUTPUT);//Pino 13 para o LED
  pinMode(2, OUTPUT); //Pino 2 do Arduino ligado ao segmento A
  pinMode(3, OUTPUT); //Pino 3 do Arduino ligado ao segmento B
  pinMode(4, OUTPUT); //Pino 4 do Arduino ligado ao segmento C
  pinMode(5, OUTPUT); //Pino 5 do Arduino ligado ao segmento D
```

```
  pinMode(6, OUTPUT); //Pino 6 do Arduino ligado ao segmento E
  pinMode(7, OUTPUT); //Pino 7 do Arduino ligado ao segmento F
  pinMode(8, OUTPUT); //Pino 8 do Arduino ligado ao segmento G
  pinMode(9, OUTPUT); //Pino 9 do Arduino ligado ao segmento PONTO
  ponto(0);  // Inicia com o ponto desligado
}
// método para acionar o ponto do display
void ponto(byte p)   //aciona o ponto{
  digitalWrite(9, p);
}
//método que configura o display
void acionaDisplay(byte digito)  //aciona o display{
  byte pin = 2;// primeira porta da sequencia
  //Percorre o array ligando os segmentos correspondentes ao digito
  for (byte cont = 0; cont < 7; ++cont){
    digitalWrite(pin, matrizDisplay[digito][cont]);
    ++pin;
  }
  ponto(1);  //Liga o ponto
  delay(100);   //Aguarda 100 milisegundos
  ponto(0);  //Desliga o ponto
}
void loop()
{
  if (Serial.available() > 0) {
    x = Serial.read ();
    switch (x) {
      case 'a': //giro para direita
            motor.attach(pMotor);
            motor.write(180);
            acionaDisplay(5);
             break;
      case 'b': //giro para esquerda
             motor.attach(pMotor);
             motor.write(0);
             acionaDisplay(6);
             break;
      case 'c': motor.attach(pMotor);
             motor.write(90);
             acionaDisplay(7);//giro lento
             break;
      case 'd': tone(pBuzzer, 2500, 1000);
```

```
            acionaDisplay(3);
             break;
      case 'e': tone(pBuzzer, 500, 1000);
            acionaDisplay(4);
            break;
      case 'f':digitalWrite(pLed, HIGH);
            acionaDisplay(1);
            break;
      case 'g':digitalWrite(pLed, LOW);
            acionaDisplay(2);
            break;
            }
else {
ponto(1);  //Liga o ponto
   delay(100);   //Aguarda 100 milisegundos
   ponto(0);  //Desliga o ponto
   delay(100);   //Aguarda 100 milisegundos
  }
delay(100);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
PROTÓTIPO PONTE
/*
Simples programa que envia dados pela porta serial
as ligações estão nos pinos 12 e 13 para os ledes
A aplaca APC220 está na pósição RX para RX e Tx para Tx.
*/
int lRed  = 12;
int lGreen = 13;
void setup() {
Serial.begin(9600);
pinMode(lRed, OUTPUT);
pinMode(lGreen, OUTPUT);
}
void  loop() {
  digitalWrite(lGreen, HIGH);
  digitalWrite(lRed, HIGH);
  delay(10);
  digitalWrite(lRed, LOW);
  delay(3000);}
```

APÊNDICE C - Código do Aplicativo - Activity Principal

```
/*
 * Autor: Célia da Silva Morais
 * Activity  - Tela Principal
 * Apresenta dois controles joysticks
 * ====================================================
 * A conexão USB utiliza a bilbioteca FTDriver
 * disponível em:
 * https://codeload.github.com/ksksue/FTDriver/zip/master
 * ====================================================
 * o layout dos controles tem base em parte do projeto  mobile anarchy
 * O projeto mobile anarchy está disponível em;
 * https://code.google.com/p/mobile-anarchy-widgets/wiki/JoystickView
 * ====================================================
 * */
package com.example.joysdroid_app;[...]
public class JoysdroidActivity extends Activity {
                FTDriver mySerial;// [FTDriver] Object
                String writeBuffer = null;
                TextView textoRespostaControle;// resposta do controle
                ImageButton abrirConexao, encerrar, voltar;
                ImageView help, conect, contTitle;
                JoystickView joystick1;
                JoystickView joystick2;
                boolean conexão = false;
                private static final String ACTION_USB_PERMISSION =
                        "com.example.joysdroid_app.USB";
            @Override
            protected void onCreate(Bundle savedInstanceState) }[...]
                        //cria instancia do objeto
        mySerial = new FTDriver((UsbManager)getSystemService(Context.USB_SERVICE));
         PendingIntent permissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(
              ACTION_USB_PERMISSION), 0);
        mySerial.setPermissionIntent(permissionIntent);
            }
            @Override
    public void onDestroy() [...]
            public void click(View v)[...]
             public void conectar(View view) {
                    if(mySerial.begin(FTDriver.BAUD9600)) // iniciar conexão {
```

```
            conexão  = true;
            abrirConexao.setEnabled(false);//deixa o botão sem acesso
                  encerrar.setEnabled(true);//permite clicar no botao de encerrar conexão
           //avisa que a conexão esta estabelecida
                  Toast.makeText(this, "CONEXÃO ESTABELECIDA",
                     Toast.LENGTH_SHORT).show();
          //deixa as imagens de ajuda invisiveis
                  help.setVisibility(View.INVISIBLE);
            conect.setVisibility(View.INVISIBLE);
            contTitle.setVisibility(View.VISIBLE);// mostra o título
        }
        else  {
            conexão = false;
            //avisa que a conexão não foi estabelecida
            Toast.makeText(this, "CONEXÃO FALHOU", Toast.LENGTH_SHORT).show();
        }
    }
    public void desconectar(View view)  {
        mySerial.end();//encerra a conexão
        conexão = false;
        abrirConexao.setEnabled(true);//deixa o botao de conectar clicável
                    contTitle.setVisibility(View.INVISIBLE);//deixa o titulo invisivel
                    //avisa que a conexão encerrou
        Toast.makeText(this, "DESCONECTADO", Toast.LENGTH_SHORT).show();
        //deixa as imagens de ajuda visiveis
        help.setVisibility(View.VISIBLE);
        conect.setVisibility(View.VISIBLE);
    }
//controle direita
    private JoystickMovedListener _listener1 = new JoystickMovedListener()   {
              @Override
              public void OnMoved(int posX, int posY)   {
                    //delimita as posições do joystick
                    int posicao = 0;
                    if((posY == 10) && (posX > -10 && posX < 10)){
                              posicao = 2;//pra baixo
                    }
                    else if((posY == -10) && (posX > -10 && posX < 10)){
                              posicao = 1; // pra cima
                    }
                    else if((posX == -10) && (posY > -10 && posY< 10)){
                               posicao = 3; //esquerda
```

```
                }
                else if((posX == 10) && (posY > -10 && posY< 10)){
                        posicao = 4; //direita
                }
                if(conexão == true){
                        switch(posicao){// envia dados
                        case 1: textoRespostaControle.setText("GIRO RÁPIDO");
                                        writeBuffer = "a";
                                        mySerial.write(writeBuffer.getBytes());
                                        break;
                        case 2: textoRespostaControle.setText("GIRO LENTO");
                                writeBuffer = "c";
                                mySerial.write(writeBuffer.getBytes());
                                break;
                        case 3: textoRespostaControle.setText("GIRO - ESQUERDA");
                                        writeBuffer = "b";
                                        mySerial.write(writeBuffer.getBytes());
                                        break;
                        case 4: textoRespostaControle.setText("GIRO - DIREITA");
                                        writeBuffer = "a";
                                        mySerial.write(writeBuffer.getBytes());
                                        break;
                        default:textoRespostaControle.setText("***");
                        }
                }
                else if(conexão == false){
                        switch(posicao){//se não há conexão, o texto de resposta recebe somente
                                                        //simbolos de indicação
                        case 1:textoRespostaControle.setText("+");break;
                        case 2: textoRespostaControle.setText("-");break;
                        case 3:textoRespostaControle.setText("<<");break;
                        case 4:textoRespostaControle.setText(">>");break;
                        default:textoRespostaControle.setText("*");
                        }
                }
        }
        @Override
        public void OnReleased
};
//controle esquerda
private JoystickMovedListener _listener2 = new JoystickMovedListener() [...]}
```

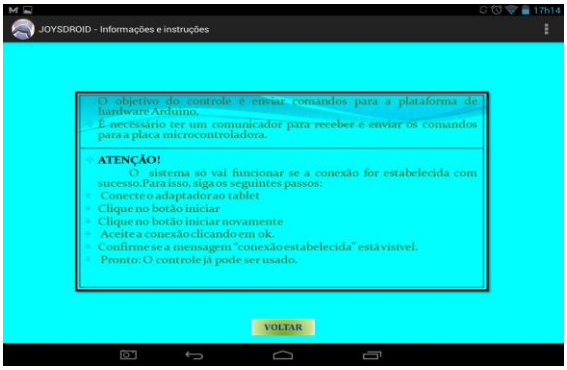
APÊNDICE D - Código da Aplicação para Desktop

```
public static void main(String[] args) {
            final Main teste = new Main();
            //abrindo comunicacao serial
            SerialInterface si = new SerialInterface("COM5", 9600);
            teste.vrText3.setText("Porta encontrada "+"\n"+" Conexão
estabelecida");
            //registrando AÇÃO de leitura que irá simplesmente ESCREVER
os dados lidos
            si.read( new SerialReadAction() {
           public void read(byte b) {
               System.out.print((char)b);
           }
                  @Override
                  public void read(String t) {
                        // TODO Auto-generated method stub

                  }});
            while (true) {
                  byte[] data = new byte[1];
                  teste.x =  teste.a.charAt(0);
                  System.out.println(teste.x);
                  data[0] = (byte) teste.x;
                  try {
                      //enviando dados via porta serial
                      si.write(data);
                      //dormindo por 1 segundo
                      Thread.sleep(1000);
                  } catch (IOException e) {
                         e.printStackTrace();
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }

            }

          }
            @Override
            public void actionPerformed(ActionEvent arg0) {
                  // TODO Auto-generated method stub

            }
  }
```

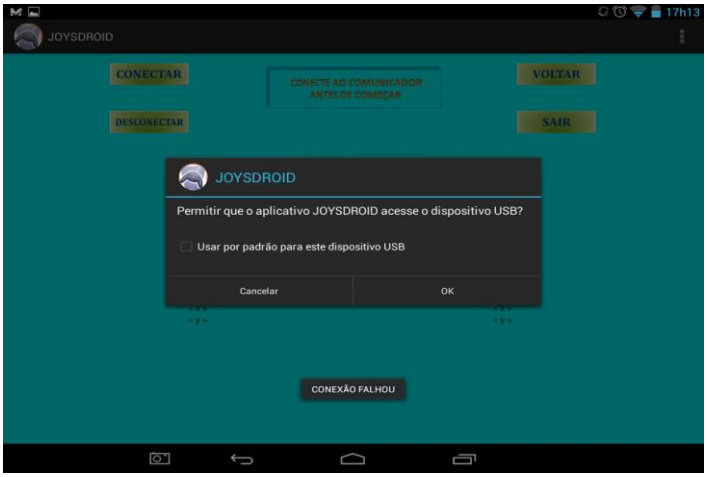
APÊNDICE E - Telas do Aplicativo

Tela Inicial - Menu

Telas de ajuda e informação



Solicita permissão para conectar



Telas do controle