Universidade Federal Fluminense
Escola de Engenharia
Curso de Graduação em Engenharia de Telecomunicações

Lúcio Folly S. Zebendo
e
João Luiz de Amorim Pereira Neto

Aplicações de *Drones* em Redes de Computadores: Utilização da plataforma *Raspberry Pi* como computador de bordo.

Niterói – RJ
2022

Lúcio Folly S. Zebendo
e
João Luiz de Amorim P. Neto

Aplicações de *Drones* em Redes de Computadores: Utilização da plataforma *Raspberry Pi* como computador de bordo.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Engenheiro de Telecomunicações.

Orientador: Prof. Dr. Alexandre Santos de la Vega
Coorientador - Prof. Dr. Lauro Eduardo Kozovits

Niterói – RJ
2022

A figura referente ao arquivo

FichaCatalografica.jpg

fornecido pela Biblioteca

deverá aparecer aqui.

ATENÇÃO: Na versão impressa, essa página deverá ficar no verso da página anterior.

Lúcio Folly S. Zebendo
e
João Luiz de Amorim P. Neto

Aplicações de *Drones* em Redes de Computadores: Utilização da plataforma *Raspberry Pi* como computador de bordo.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Engenheiro de Telecomunicações.

Aprovada em DIA de MÊS de ANO.

BANCA EXAMINADORA

Prof. Dr. Alexandre S. de la Vega - Orientador
Universidade Federal Fluminense - UFF

Prof. Dr. Lauro Eduardo Kozovits - Co-Orientador
Universidade Federal Fluminense - UFF

Prof. Alexandre S. de la Vega
INSTITUIÇÃO

Prof. Lauro Eduardo Kozovits
INSTITUIÇÃO

Niterói – RJ
2022

Resumo

Este trabalho é o fruto de um esforço conjunto entre vários discentes e docentes da UFF e iniciativa privada, todos com mútuo interesse por VANTs e suas aplicações. O projeto inicial era ideia da Equipe UFFO [Equipe UFFO,] - construir um *drone* para vigilância dos campi da UFF. O hardware a ser utilizado era um quadricóptero equipado com um sistema FPV de transmissão analógica. Posteriormente, houve-se o interesse de equipar um computador de bordo no *drone* em questão, para fazer aplicações no campo da visão computacional com processamento em tempo de voo. Para isso foi utilizado como referência o material da comunidade [Ardupilot Docs, 2021] sobre computadores de bordo [Companion Computers, 2021].

Para a escolha do computador de bordo foram levados mais em conta fatores de custo e popularidade, com isso a plataforma *Raspberry Pi* [Foundation, 2020] foi a selecionada para esse fim.

A partir da arquitetura proposta, várias aplicações além da visão computacional são possíveis. Como o computador de bordo em questão possui um sistema operacional com propósito geral baseado em *debian* [Software in the Public Interest, Inc, 2021], além de outras capacidades de Hardware como portas *GPIO* para acoplamento de sensores e atuadores e outros barramentos para conexão de periféricos, as possibilidades para esse hardware são muitas.

Com isso, a aplicação proposta nesse trabalho será a conexão desse computador de bordo à uma rede IP. Várias capacidades da plataforma *Raspberry Pi* serão exploradas nos experimentos que sucederão, além disso, serão desenvolvidas aplicações para testar o conceito de *Drones* em Redes de computadores.

Palavras-chave: Raspberry Pi. Drones. VANT. ROS. Equipe UFFO. Robótica. Ardpilot. Pixhawk

Abstract

This part is destinated to the abstract of your monograph. It must be written in the vernacular language and in an idiom of great popularization (English, French, Spanish, for example). This part should be done at last, because just after finishing the work it will be possible an overall understanding of it. The abstract should not bring any further information, it is just the summary of the relevants aspects of the monograph, such as work gender, finality, methodology, results and conclusions. It must be written impersonally, to possess an extension from 150 to 500 words typed in simple space and in only one paragraph. It must be followed by the keywords of your monograph.

Keywords: Monograph. LaTeX. Hints.

Espaço reservado para a dedicatória.

Agradecimentos

Espaço reservado para os agradecimentos. Agradecemos ao Prof. Alexandre pela orientação durante a nossa estadia no grupo do PET-Tele. Através, da nossa participação no grupo foi possível termos um primeiro contato com as plataformas de desenvolvimento Arduino e *Raspberry Pi* [Foundation, 2020], além de outras atividades que contribuíram para nosso desenvolvimento profissional. Agradecemos ao Prof. Raul pelo acolhimento na *Equipe UFFO* e pelo incentivo ao estudo e desenvolvimento de VANTs. Agradecemos ao Prof. Lauro Eduardo pelo direcionamento durante a elaboração desse projeto de TCC. Agradecemos à Equipe UFFO e especialmente ao Raphael Miranda, pelo apoio no desenvolvimento do drone quadcóptero utilizado nos experimentos desse trabalho. Agradecemos à Empresa 6DDrones e especialmente ao Fabio, pelo apoio com infraestrutura e equipamentos para o desenvolvimento desse projeto. Agradecemos à comunidade Arducopter pelo grande esforço colaborativo em desenvolver varias das ferramentas utilizadas nesse projeto. Agradeço à Larissa, minha namorada, por todo apoio, carinho e incentivo durante a minha graduação. Agradecemos aos nossos familiares pela cobrança e incentivo nos estudos.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Atrelado à popularização da tecnologia, o barateamento dos *drones* tornaram-nos capazes de serem usados para além do âmbito militar, ganhando espaço nas industrias cinematográficas, esportivas e do entretenimento, fora para o uso pessoal.
Somado à isso, com o barateamento e miniaturização dos componentes eletrônicos nas últimas décadas, surgiram diversas plataformas de difusão de estudos sobre possíveis aplicações é, portanto, uma consequência inevitável. Dessa forma, abre-se um leque de possibilidades para explorar as atividades que um VANT (Veículo Aéreo Não Tripulado) pode executar.

1.1 Motivações

O sentimento de interesse e empolgação pela tecnologia dos drones dos autores desse texto foram as forças primordiais para a concepção do trabalho que se segue. Tal interesse nasceu dentro da Universidade Federal Fluminense através da equipe UFFO [Equipe UFFO,], formada por discentes e docentes da universidade.
Com a experiência adquirida na confecção de um drone quadcóptero simples, descobriu-se a possibilidade de acoplar a este um Raspberry Pi [Foundation, 2020] como computador de bordo da aeronave, expandindo suas capacidades.
A partir desta ideia, foram realizados estudos para buscar os métodos e tecnologias mais recentes utilizados nessa configuração de drone. Felizmente, muitas dessas tecnologias utilizadas já estavam disponíveis na documentação Open Source do Ardupilot [Ardupilot Docs, 2021] para o equipamento utilizado.

1.2 Objetivo

O primeiro objetivo deste trabalho é projetar, construir e testar uma conexão de um drone a uma rede *IP*, visando facilitar a coleta de dados e o controle do drone em si. Para isso, no ambiente simulado, utilizaremos uma rede *wi-fi* para realizar o envio de dados do drone a um servidor remoto, cujo intuito é armazenar, processar e exibir informações coletadas pelo drone. O segundo objetivo, agora fora do ambiente de simulação, gira em torno do estudo da viabilidade e benefícios de uma conexão utilizando a tecnologia LTE. ampliando o ambiente de funcionamento para muito além do alcance do *wi-fi*. *IP*.

1.3 Resultados esperados

- Criação de um ambiente de simulação para a validação dos experimentos propostos.
- Um sistema de controle e navegação para o drone via rede IP.
- Um drone capaz de realizar comunicação de dados numa rede IP.

1.4 Trabalhos correlacionados

- Artigos de Drones LTE [Lassi Sundqvist, 2015]

1.5 Organização do documento

Resumir os seguintes pontos:

1. Introducao teorica
2. Definições
3. Arquitetura Proposta
4. Metodologia

Capítulo 2

Introdução teórica

2.1 Definições iniciais

2.1.1 Drones e VANTs

Drones são sistemas aéreos automatizados que incluem tanto os VANTs - Veículos Aéreos Não Tripulados, capazes de voar por milhares de kilometros, quanto drones pequenos que voam em locais confinados. São veículos aéreos não tripulados, automatizados ou controlados remotamente, podem carregar diversos tipos de carga e fazer vários tipos de missões.
Devido a avanços nas tecnologias envolvidas na fabricação, nos sistemas de navegação, nos sensores e sistemas de armazenamento de energia, uma grande variedade de tipos de drones surgiram nas últimas décadas.

2.1.2 Classificação de VANTs

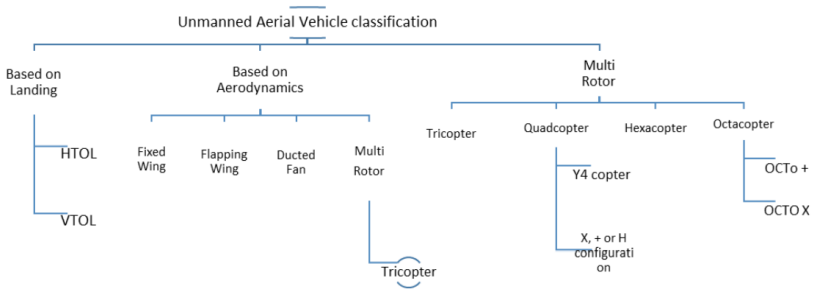

Figura 2.1: Classificação de VANTs por tipo de pouso, aerodinâmica e quantidade de rotores.

Como mostrado no organograma da figura a cima 2.1, as categorizações mais comuns para os VANTs são obtidas pelos atributos "tipo de pouso", "aerodinâmica"e "quan-

tidade de rotores".

O tipo de aerodinâmica responsável pela sustentação das aeronaves no ar é bastante diversificado. Uma categoria bem comum de drones são os VANTs de asa fixa, que semelhantemente aos aviões, tiram proveito da geometria das asas e da velocidade gerada por um ou mais motores para obterem sustentação.

A classificaçao de aterrisagem, com a nomenclatura VTOL e HTOL diz respeito apenas aos VANTs de asa fixa visto que apenas esses tipos de VANTs tiram proveito tanto da aterrisagem vertical quanto da autonomia proveniente do vôo horizontal com asas fixas.

2.1.3 Componentes dos VANTs de Asa Rotativa

Os componentes dos drones de asa rotativa variam conforme o tipo de missão. Alguns dos componentes proncipais podem ser listados abaixo:

1. Frame
2. Motores e hélices
3. Escs
4. Controladora de voo
5. Bateria e Placa distribuidora de energia

Frame

O Frame de um drone é a base estrutural na qual todos os outros componentes são fixados, ele geralmente é feito de um material leve e resistente como fibra de carbono e outros polímeros plásticos.

Bateria e Placa distribuidora de energia

A bateria é a fonte de energia do drone, geralmente são utilizadas baterias Li-Po (Lítio-Polímero) pela sua boa relação massa/desempenho enérgetico, essas baterias possuem uma nomenclatura específica com baase na quantidade de células <explicar melhor as baterias>.

Motores e hélices

Os motores do drone são os responsávei por converter a energia elétrica proveniente das baterias em energia cinética. Geralmente os motores utilizados são motores de

corrente contínua não escovados (*brushless DC*) <colocar as características físicas desse tipo de motor>. A partir da corrente elétrica fornecida ao motor ele movimenta a hélice, que devido à sua aerodinamica gera uma força resultante para cima no drone, a qual é denominada empuxo.

Escs

Os Escs, ou Eletronic Speed Controlers são os componentes eletrônicos responsáveis por controlar a velocidade dos motores. Eles são comandados pela controladora de voo, geralmente a partir de sinais PWM <Falar sobre PWM>. Assim como o PWM é utilizado para comandar os ESCs, os ESCs utilizam essa modulação para chavear o circuito entre os motores e a bateria controlando assim a velocidade dos motores.

Controladora de Voo

A controladora de voo é um microcontrolador com software embarcado dedicado às funções básicas da aeronave. Ela possui sensores inerciais embutidos para comandar a dinâmica de voo da aeronave, além de possuir interface eletrônica com outros componentes como módulos gps, módulos receptores de rádio, módulos de telemetria entre outros. Sem a controladora de voo, seria uma tarefa humanamente impossível controlar o voo da aeronave, visto que a mesma depende de um feedback contínuo e instantâneo dos parâmetros inerciais.

2.1.4 Ardupilot

blalbalalba

2.1.5 UART

O controlador UART (Universal Asynchronous Receiver/Transmitter) é um componente utilizado para fazer a comunicação serial entre dispositivos. Este componente geralmente apresentado como um circuito integrado, consegue transmitir bits de dados de forma síncrona e assíncrona entre dispositivos.

<descrever funcionamento>

2.1.6 Telemetria

A telemetria é uma funcionalidade indispensável nos drones mais modernos, ela permite receber informações dos sensores do drone em tempo real como altitude, velocidade, gps, temperatura, informações da quantidade de carga na bateria dentre outras.

Essa transmissãode dados geralmente é feita por um módulo dedicado a isso ou então em módulos receptores de rádio que já possuem essa funcionalidade integrada.

2.1.7 Protocolo Mavlink

O Mavlink é um protocolo leve para comunicação com drones e também entre os componentes embarcados dos mesmos. Ele segue uma implementação moderna da arquitetura publisher-subscriber e point to point

MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with retransmission.

Messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects build on top of this definition).

Code generators create software libraries for specific programming languages from these XML message definitions, which can then be used by drones, ground control stations, and other MAVLink systems to communicate. The generated libraries are typically MIT-licensed, and can therefore be used without limits in any closed-source application without publishing the source code of the closed-source application.

Key Features Very efficient. MAVLink 1 has just 8 bytes overhead per packet, including start sign and packet drop detection. MAVLink 2 has just 14 bytes of overhead (but is a much more secure and extensible protocol). Because MAVLink doesn't require any additional framing it is very well suited for applications with very limited communication bandwidth. Very reliable. MAVLink has been used since 2009 to communicate between many different vehicles, ground stations (and other nodes) over varied and challenging communication channels (high latency/noise). It provides methods for detecting packet drops, corruption, and for packet authentication. Many different programming languages can be used, running on numerous microcontrollers/operating systems (including ARM7, ATMega, dsPic, STM32 and Windows, Linux, MacOS, Android and iOS). Allows up to 255 concurrent systems on the network (vehicles, ground stations, etc.) Enables both offboard and onboard communications (e.g. between a GCS and drone, and between drone autopilot and MAVLink enabled drone camera).

2.1.8 Computadores de Bordo

Companion Computers can be used to interface and communicate with ArduPilot on a flight controller using the MAVLink protocol. By doing this your companion compu-

ter gets all the MAVLink data produced by the autopilot (including GPS data) and can use it to make intelligent decisions during flight. For example, “take a photo when the vehicle is at these GPS co-ordinates”, gather and pre-process information from advanced sensors or actuate lights, auxiliary servos or any other interfaces.

There are two major parts to Companion Computers - hardware and software.

The Companion Computer hardware refers to the specific computer hardware being used. This is typically a small ARM-based Single Board Computer. Specific tutorials for popular Companion Computer hardware are listed below.

Arduino family LYCHEE (Cube Carrier Board for Raspberry Pi Compute Module) NVidia TX1 NVidia TX2 ODroid Raspberry Pi The Companion Computer software refers to the programs and tools that run on the Companion Computer. They will take in MAVLink telemetry from the Flight Controller and can route and process the telemetry data. Specific tutorials for popular Companion Computer software tools/suites are listed below.

APSync DroneKit FlytOS Maverick ROS Rpanion-server

2.1.9 ROS - Robot Operative System

Robot Operating System (ROS or ros) is an open-source robotics middleware suite. Although ROS is not an operating system (OS) but a set of software frameworks for robot software development, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post, and multiplex sensor data, control, state, planning, actuator, and other messages. Despite the importance of reactivity and low latency in robot control, ROS is not a real-time operating system (RTOS). However, it is possible to integrate ROS with real-time code.[3] The lack of support for real-time systems has been addressed in the creation of ROS 2,[4][5][6] a major revision of the ROS API which will take advantage of modern libraries and technologies for core ROS functions and add support for real-time code and embedded system hardware.

Software in the ROS Ecosystem[7] can be separated into three groups:

language-and platform-independent tools used for building and distributing ROS-based software; ROS client library implementations such as roscpp,[8] rospy,[9] and roslisp;[10] packages containing application-related code which uses one or more ROS client libraries.[11] Both the language-independent tools and the main client libraries (C++, Python, and Lisp) are released under the terms of the BSD license, and as such are open-source software and free for both commercial and research use. The majority of

other packages are licensed under a variety of open-source licenses. These other packages implement commonly used functionality and applications such as hardware drivers, robot models, datatypes, planning, perception, simultaneous localization and mapping, simulation tools, and other algorithms.

The main ROS client libraries are geared toward a Unix-like system, primarily because of their dependence on large collections of open-source software dependencies. For these client libraries, Ubuntu Linux is listed as "Supported"while other variants such as Fedora Linux, macOS, and Microsoft Windows are designated "experimental"and are supported by the community.[12] The native Java ROS client library, rosjava,[13] however, does not share these limitations and has enabled ROS-based software to be written for the Android OS.[14] rosjava has also enabled ROS to be integrated into an officially supported MATLAB toolbox which can be used on Linux, macOS, and Microsoft Windows.[15] A JavaScript client library, roslibjs[16] has also been developed which enables integration of software into a ROS system via any standards-compliant web browser.

2.2 Arquitetura do sistema proposto

Com todas as tecnologias devidamente introduzidas, deseja-se criar um sistema composto pelas mesmas e explorar as suas capacidades. Para isso, utilizaremos um drone com controladora ardupilot e computador de bordo capaz de se conectar com uma rede de computadores. O software embarcado será desenvolvido utilizando ROS e um API será desenvolvida em django para centralizar se conectar ao drone via rede.

Capítulo 3

Metodologia

3.1 Pesquisa

3.2 Objetivos do método aplicado

A metodologia aplicada possui objetivo descritivo e exploratório. Com o sistema devidamente construído, faremos experimentos que comprovam algumas possibilidades de uso para o equipamento.

3.3 Abordagem

adotar-se-á uma análise qualitativa dos resultados obtidos a partir da implementação do novo sistema, em observância aos seguintes questionamentos:

1. Com o sistema desenvolvido será possível ao professor, operador do sistema. controlar o experimento remotamente?
2. Com o sistema desenvolvido a turma da disciplina conseguirá acompanhar o experimento em um ambiente diferente do laboratório?
3. pergunta avaliativa 3
4. pergunta avaliativa 4

3.4 Descrição do problema

Capítulo 4

Detalhamento da solução

4.1 Série de computadores Raspberry Pi

O Raspberry Pi é uma série de computadores de placa única e tamanho reduzido, que recebem a denominação SoC (*System on Chip*) [Members, 2019b] à qual são conectados os seguintes dispositivos: monitor, teclado, e mouse.

Desenvolvido no Reino Unido pela Raspberry Pi Foundation tendo, como principais objetivos, contribuir para inclusão digital, promoção de ensino básico em ciência da computação e empoderamento social. Uma alternativa de ensino com baixo custo para escolas e estudantes [Members, 2019a]. A Tabela 4.1 apresenta as especificações técnicas do modelo Raspberry Pi Revision 2 - Element 14, utilizado pelo grupo PET-Tele.

Chip	Broadcom BCM2835 SoC Full HD Processador de Aplicações Multimídia
CPU	700 Mhz ARM1176JZ-F Processador de Baixa Potência de aplicações
GPU	Dois Núcleos, VideoCore IV, Co-Processador de Multimídia
Memória	512MB SDRAM
Ethernet	Onboard 10/100 Ethernet com conector RJ-45
USB 2.0	Dois Conectores USB 2.0
Saída de Vídeo	HDMI e Composição RCA (PAL e NTSC)
Saída de Áudio	Conector 3.5 mm ou HDMI
Armazenamento	SD, MMC, SDIO Card Slot
Dimensões	8.6 cm X 5.4 cm X 1.7 cm

Tabela 4.1: Especificações técnicas do modelo Raspberry Pi Revision 2 - Element14.

Na Figura 4.1 é mostrado uma fotografia do modelo. Dentre os 25 Pinos presentes neste Raspberry, 17 podem ser usados como entradas ou saídas de uso geral, 5 como terminais comuns (GND), 2 como fontes de tensão de valor +5V e 2 como fontes de tensão de valor +3.3V.

A Tabela 4.2 reúne uma descrição de todos os pinos.

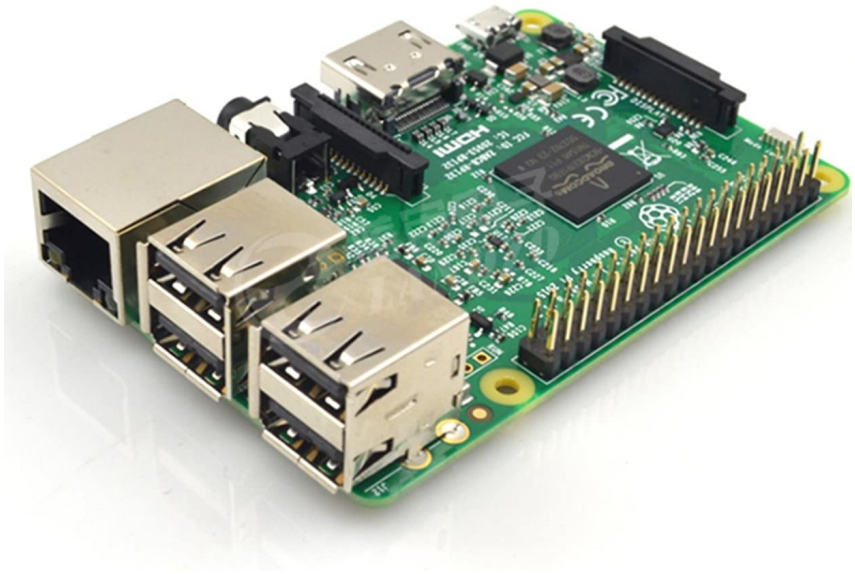

Figura 4.1: Raspberry Pi Revision 2.0.

N	**Descrição**	**N**	**Descrição**
1	Saída de +3.3V	14	Terminal Comum GND
2	Saída de +5V	15	GPIO22
3	GPIO2 | Pull-Up | I2C-SDA	16	GPIO23
4	Saída de +5V	17	3V3
5	GPIO3 | Pull-Up | I2C-SCE	18	GPIO24
6	Terminal Comum GND	19	GPIO10 | SPI | MOSI
7	GPIO4	20	Terminal Comum GND
8	GPIO14 | UART | TXD	21	GPIO9 | SPI | MISO
9	Terminal Comum GND	22	GPIO25
10	GPIO15 | UART | RXD	23	GPIO11 | SPI | CLK
11	GPIO17 | UART-RTS	24	GPIO8 | SPI | CE0
12	GPIO18 | PWM	25	Terminal Comum GND
13	GPIO27	26	GPIO7 | SPI | CE1

Tabela 4.2: Descrição dos pinos do Raspberry Pi Rev 2.0 - Element 14.

Capítulo 5

Resultados

apresentação de resultados (numéricos e/ou gráficos) de cálculos e/ou de simulações, requeridos na especificação do trabalho.

Capítulo 6

Conclusão

Hipótese testada Se realizarmos o experimento em outro ambiente com maior capacidade para que todos possam assistir de forma homogênea atenderemos a demanda da turma, teremos um aprendizado mais efetivo e melhoria no tempo dedicado a esse experimento.

Responda as perguntas avaliativas para definir a conclusão

1. Com o sistema desenvolvido será possível ao professor usuário controlar o experimento proposto remotamente?
2. pergunta avaliativa 2
3. pergunta avaliativa 3
4. pergunta avaliativa 4

A presente monografia detalha o processo de pesquisa, projeto e construção de um sistema para atuação em uma planta didática localizada no Laboratório de Drenagem Irrigação e Saneamento [LaDISan, 2019].

Utilizando, como base, o documento **Uso de Sensores e Acesso Remoto para a Realização de Aula Prática Sobre Reservatórios de Detenção Aplicados à Drenagem urbana** [Paiva and Milhomem, 2018] e o artigo **Realização de Aula Prática Remota a Partir de Laboratório Equipado com Modelo Físico Sobre Detenção de Água de Chuva** [Paiva et al., 2018], publicado no COBENGE 2018 [ABENGE, 2020], foi desenvolvido um conjunto de melhorias e novos processos que permitem monitoramento e maior e melhor atuação do "usuário" (computador remoto localizado na sala de aula) na planta didática operando no Laboratório LaDISan [LaDISan, 2019]. O conjunto de melhorias compreende uma câmera com controle remoto, um servomotor acoplado a um registro de admissão de água, uma interface de acesso e um dispositivo Raspberry Pi operando como controlador do servomotor e computador auxiliar para o Arduino.

Os resultados foram satisfatórios no que tange aos objetivos propostos. Foram realizadas duas aulas no ano de 2019, uma no primeiro semestre letivo e outra no segundo semestre.

O sistema ainda não contempla uma solução fechada pronta para comercialização conforme foi citado no Capítulo 3. Ainda se faz necessário melhorias e correções que podem ser implementadas conforme novas demandas levantadas por alunos da disciplina e o professor Dario Sobrenome Sobrenome. As demandas podem ser implementadas pelos futuros bolsistas do grupo PET-Tele. Complementarmente a este documento existe um relatório técnico com detalhamento das etapas de construção, disponível no *website* do Grupo PET-Tele [Neto and Miranda, 2021]

Capítulo 7

Sugestões para trabalhos futuros

Em observância ao caráter amplo e diverso dos conceitos aqui utilizados, diversas vertentes de trabalhos futuros podem ser identificadas. Tais vertentes, assim como os trabalhos individuais em cada uma das vertentes, podem ser listados e resumidos.

1. Vertente Técnica Funcional
   (a) Novas interfaces de usuário com mais opções
   (b) Desenvolvimento de novas curvas hidrográficas
2. Vertente Facilidade de Uso
   (a) Encapsulamento de hardware em produto final
   (b) Simplificação do configuração e uso para usuário final.
3. Vertente casos de testes
   (a) Coleta de indicadores para medir o grau de satisfação da turma com o sistema.
   (b) Aumentar o número de testes a fim de identificar falhas.

Vale ressaltar dentro dessas vertentes a necessidade de realização de mais testes com o sistema em operação como forma de garantir que as funcionalidades propostas estejam com o funcionamento correto.]

Para isso, é de fundamental importância a elaboração de novos casos de uso e simulações teste, bem como a implementação de atualizações otimizadas dos programas desenvolvidos. Uma pesquisa avaliando os parâmetros funcionais qualitativos da interface proposta na Seção **??** pode servir de suporte para as exigências aqui formuladas.

Referências Bibliográficas

[ABENGE, 2020] ABENGE (2020). **URL do COBENGE**. Disponível em: "`http://www.abenge.org.br/cobenge.php`". Acesso em: 25/01/2021.

[Ardupilot Docs, 2021] Ardupilot Docs (2021). **Ardupilot Docs**. Disponível em: "`https://ardupilot.org/dev/#welcome-to-the-ardupilot-development-site`". Acesso em: 25/01/2021.

[Companion Computers, 2021] Companion Computers (2021). **Companion Computers**. Disponível em: "`https://ardupilot.org/dev/docs/companion-computers.html`". Acesso em: 25/01/2021.

[Equipe UFFO,] Equipe UFFO. **Equipe UFFO**. Disponível em: "`https://uffoequipe.github.io/website/uffo/`". Acesso em: 25/01/2021.

[Foundation, 2020] Foundation, R. P. (2020). **URL da plataforma Raspberry Pi**. Disponível em: "`https://www.raspberrypi.org/`". Acesso em: 25/01/2021.

[LaDISan, 2019] LaDISan (2019). **URL do Laboratório de Drenagem, Irrigação e Saneamento Ambiental (TER/UFF)**. Disponível em: "`http://www.uff.br/?q=setor/laboratorio-de-drenagem-irrigacao-e-saneamento-ambiental-ladisan`". Acesso em: 25/01/2021.

[Lassi Sundqvist, 2015] Lassi Sundqvist (2015). ***cellular controlled drone experiment: evaluation of network requirements***. Disponível em: "`https://aaltodoc.aalto.fi/bitstream/handle/123456789/19152/master_Sundqvist_Lassi_2015.pdf?sequence=1&isAllowed=y`". Acesso em: 25/01/2021.

[Members, 2019a] Members, W. (2019a). **URL do termo Raspberry Pi**. Disponível em: "`https://pt.wikipedia.org/wiki/Raspberry_Pi`". Acesso em: 25/01/2021.

[Members, 2019b] Members, W. (2019b). **URL do termo *System on a chip* (SoC)**. Disponível em: "`https://pt.wikipedia.org/wiki/System-on-a-chip`". Acesso em: 25/01/2021.

[Neto and Miranda, 2021] Neto, J. L. A. P. and Miranda, R. (2021). **Novas implementações para automatização e acesso remoto no sistema de reservatórios de detenção aplicados à drenagem urbana do LaDISan/TCE/UFF utilizando a plataforma Raspberry Pi**. Disponível em: "`http://www.telecom.uff.br/pet/petws/index.php?pagina=downloads/projetos`". Acesso em: 25/01/2021.

[Paiva and Milhomem, 2018] Paiva, L. M. M. and Milhomem, M. C. (2018). **Uso de sensores e acesso remoto para a realização de aula prática sobre reservatórios de detenção aplicados à drenagem urbana**. *UFF*, page 70. Disponível em: "`https://app.uff.br/riuff/handle/1/7080`".

[Paiva et al., 2018] Paiva, L. M. M., Milhomem, M. C., and de la Vega, A. S. (2018). **Realização de Aula Prática Remota a Partir de Laboratório Equipado com Modelo Físico sobre Detenção de Água de Chuva**. *COBENGE*, page 10. Disponível em: "`http://www.telecom.uff.br/pet/petws/downloads/artigos/cobenge_2018/COBENGE18_00063_00001258_Aula_Remota_LaDISan_2018_07_09.pdf`".

[Software in the Public Interest, Inc, 2021] Software in the Public Interest, Inc (2021). **Debian OS**. Disponível em: "`https://www.debian.org/`". Acesso em: 25/01/2021.

Apêndice A

Apêndice 1

Segue abaixo um guia de instalação completo dos *Softwares* de simulação e suas respectivas dependências para Ubuntu 20.04 LTS.

A.1 Intalando o ArduPilot e MavProxy

A.1.1 Clonando o repositório *git* para sua máquina

```
$ cd ~
$ sudo apt install git
$ git clone https://github.com/ArduPilot/ardupilot.git
```

A.1.2 Instalando as dependências e recompilando o perfil

```
$ cd ardupilot/Tools/environment_install/install-prereqs-
   ubuntu.sh -y
$ . ~/.profile
```

A.1.3 Mudando para a *branch* do ArduCopter

```
$ git checkout Copter-4.0.4
$ git submodule update --init --recursive
```

A.1.4 Rodando SITL (*Software In The Loop*)

```
$ cd ~/ardupilot/ArduCopter
$ sim_vehicle.py -w
```

A.2 Instalando o Gazebo e o *plugin* do ArduPilot

A.2.1 Atualizando a lista de fontes para *download*

```
$ sudo sh -c 'echo "deb http://packages.osrfoundation.org/
   gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/
   sources.list.d/gazebo-stable.list'
$ wget http://packages.osrfoundation.org/gazebo.key -O - |
   sudo apt-key add -
$ sudo apt update
```

A.2.2 Instalando o *plugin* do Gazebo para comunicação com o ArduPilot

```
$ cd ~
$ git clone https://github.com/khancyr/ardupilot_gazebo.git
$ cd ardupilot_gazebo
$ mkdir build
$ cd build
$ cmake ..
$ make -j4
$ sudo make install
$ echo 'source /usr/share/gazebo/setup.sh' >> ~/.bashrc
$ echo 'export GAZEBO_MODEL_PATH=~/ardupilot_gazebo/models'
   >> ~/.bashrc
$ . ~/.bashrc
```

A.2.3 Executando a simulação

```
No primeiro terminal do linux rode o Gazebo:
$ gazebo --verbose ~/ardupilot_gazebo/worlds/
   iris_arducopter_runway.world
```

```
No segundo terminal do linux rode o SITL:
$ cd ~/ardupilot/ArduCopter/
$ sim_vehicle.py -v ArduCopter -f gazebo-iris --console
```

A.3 Instalando o *ROS* e o configurando o *Catkin*

A.3.1 Atualizando a lista de fontes para *download*

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(
   lsb_release -sc) main" > /etc/apt/sources.list.d/ros-
   latest.list'
$ sudo apt install curl
$ curl -s https://raw.githubusercontent.com/ros/rosdistro/
   master/ros.asc | sudo apt-key add -
$ sudo apt update
```

A.3.2 Instalando o *ROS*

```
$ sudo apt install ros-noetic-desktop-full
```

A.3.3 Configurando o ambiente

```
$ source /opt/ros/noetic/setup.bash
$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
$ echo "source /opt/ros/noetic/setup.zsh" >> ~/.zshrc
$ source ~/.zshrc
```

A.3.4 Instalando as dependências para os pacotes dos *ROS*

```
$ sudo apt install python3-rosdep python3-rosinstall python3-
   rosinstall-generator python3-wstool build-essential
$ sudo apt install python3-rosdep
$ sudo rosdep init
$ rosdep update
```

A.3.5 Configurando o *Catkin*

```
$ sudo apt-get install python3-wstool python3-rosinstall-
   generator python3-catkin-lint python3-pip python3-catkin-
   tools
$ pip3 install osrf-pycommon
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws
$ catkin init
```

A.3.6 Instalando as dependências para o *Catkin*

```
$ sudo apt-get install python3-wstool python3-rosinstall-
   generator python3-catkin-lint python3-pip python3-catkin-
   tools
$ pip3 install osrf-pycommon
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws
$ catkin init
```

A.3.7 Instalando as *MAVROS* e *MAVLink*

```
$ cd ~/catkin_ws
$ wstool init ~/catkin_ws/src
$ rosinstall_generator --upstream mavros | tee /tmp/mavros.
   rosinstall
$ rosinstall_generator mavlink | tee -a /tmp/mavros.
   rosinstall
$ wstool merge -t src /tmp/mavros.rosinstall
$ wstool update -t src
$ rosdep install --from-paths src --ignore-src --rosdistro `
   echo $ROS_DISTRO` -y
$ catkin build
```

A.3.8 Atualizando o arquivo *.bachrc*

```
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

A.4 Instalando as dependências geográficas e clonando o repositório de simulação do *Intelligent Quads*

A.4.1 Instalando as dependências geográficas

```
$ sudo ~/catkin_ws/src/mavros/mavros/scripts/
   install_geographiclib_datasets.sh]
```

A.4.2 Clonando pacote ROS de simulação do *Intelligent Quads*

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/Intelligent-Quads/iq_sim.git
$echo "GAZEBO_MODEL_PATH=${GAZEBO_MODEL_PATH}:$HOME/catkin_ws
   /src/iq_sim/models" >> ~/.bashrc
$ cd ~/catkin_ws
$ catkin build
$ source ~/.bashrc
```

A.5 Instalando o *QGround Control*

A.5.1 Alterando permissões e instalando o *QGround Control*

```
$ sudo usermod -a -G dialout $USER
$ sudo apt-get remove modemmanager
$ wget https://s3-us-west-2.amazonaws.com/qgroundcontrol/
   latest/QGroundControl.AppImage
$ chmod +x ./QGroundControl.AppImage
$ ./QGroundControl.AppImage
```