Trabalho de Conclusão de Curso
2ª Etapa – Projeto do Sistema

SIDEL: Interface de controle de drone para inspeção de torres de distribuição de energia elétrica

Integrantes do Grupo:

Guilherme Luis Frandina
Ana Júlia Ramalho de Castro

Orientador(a): Álvaro Rogério Cantieri

Co-orientador(a): Gabriel Vinicius Canzi Candido

Pinhais
2024

1. Introdução

O sistema de geração e distribuição de energia elétrica no Brasil tem como principal base de geração a energia hidroelétrica, que requer o aproveitamento de recursos hídricos em áreas onde elas estão disponíveis geograficamente, tornando a rede muito extensa (EPE, 2023). Somado com a grande extensão do território brasileiro, essa infraestrutura se torna muito mais abrangente, tornando a transmissão e distribuição desse recurso em um grande desafio, exigindo formas de controle e de supervisão periódicas para que acidentes não ocorram (SkyFire, 2023) .

A COPEL (Companhia Paranaense de Energia), por exemplo, é a empresa responsável pelo fornecimento de energia elétrica para o estado do Paraná, possuindo mais de 200 mil quilômetros de linhas de distribuição (COPEL, 2021). Com partes dessas linhas sendo feitas através de redes aéreas, compostas por torres com cabos aéreos com revestimento isolante que passam por diversos pontos do estado.

Segundo o guia de inspeção de vants da IEEE (IEEE, 2020), na forma manual de controle de drone são necessários dois técnicos, tornando o processo de análise mais simples e seguro do que o procedimento tradicional (que era feito por meio de helicópteros próximo às torres) (AEN, 2021), visto que evita situações de risco.

Porém, nesse método de operação fica sob responsabilidade do piloto desviar de obstáculos e evitar colisões com a aeronave, além de, ter que realizar a direção do drone até o ponto de inspeção. Como consequência disso, a tarefa de inspecionar se torna difícil, mesmo com drones que apresentam sistemas de segurança - como sensores e algoritmos auxiliares - que ainda necessitam de um alto grau de habilidade do controlador, podendo levar a acidentes com a aeronave e a estrutura da torre.

Por isso, esse projeto tem como objetivo a criação de uma interface amigável para a operação do VANT (Veículo Aéreo Não Tripulado) que será desenvolvida por outro projeto que está sendo realizado em conjunto no Laboratório de Robótica e Computação Aplicada do IFPR Pinhais (LaRCA).

2. Lista de requisitos funcionais

Incluir a lista dos requisitos funcionais do sistema. Exemplo:

REF01: Uma mensagem de status deve ser mostrada na área inferior da janela (desenho da Fig.1)

REF02: A mensagem deve ser atualizada a cada 60 segundos, com tolerância de 10 segundos para mais ou para menos

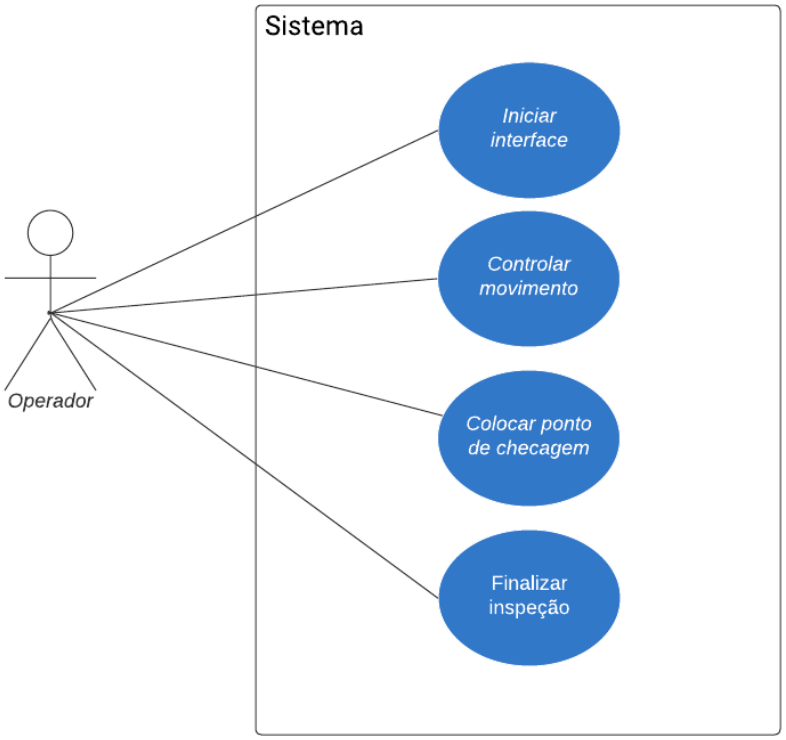
3. Lista de requisitos não funcionais

Incluir a lista dos requisitos não funcionais do sistema. Exemplo:

RNF01: O sistema deve ficar disponível por 99,5% do tempo nos dias úteis, das 6h às 22h

RNF02: Em condições de pico de uso, deve ter uma reserva de 25% de capacidade de processamento e memória

4. Diagrama de casos de uso



Fonte: Os autores (2024)

O diagrama representa os principais casos de uso da interface desenvolvida para o drone automatizado, que será utilizado na inspeção de torres de distribuição de energia elétrica. Ele detalha as ações que o operador pode realizar ao interagir com o sistema.

Ator - Operador
O ator no diagrama é o operador, que representa o usuário responsável por controlar e monitorar o sistema do drone durante a inspeção. Este ator interage diretamente com o sistema por meio da interface desenvolvida.

4.1 Casos de Uso

Iniciar Interface:
Este caso de uso descreve a ação inicial do operador ao abrir e configurar o sistema para começar a operação. Envolve carregar a interface com o drone preparado para a inspeção.

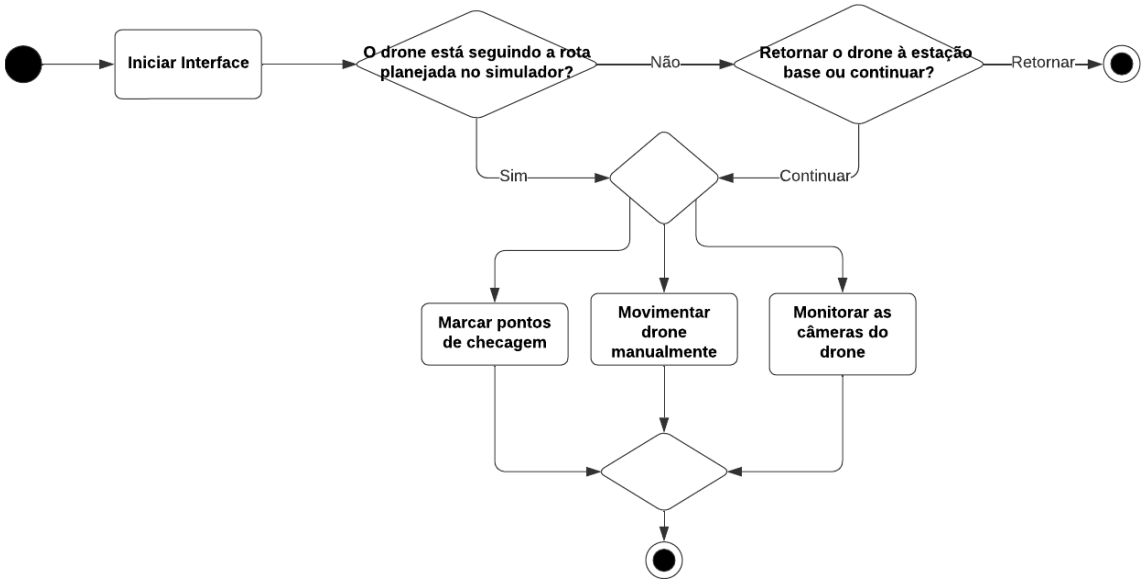
Controlar Movimento:
Permite ao operador ajustar o movimento do drone durante a inspeção. Isso inclui comandos básicos como mover para frente, para trás, subir, descer e ajustar a direção.

Colocar Ponto de Checagem:
Este caso de uso cobre a funcionalidade de definir pontos específicos para inspeção ou coleta de dados. O operador pode marcar no sistema os locais que exigem maior atenção ou que devem ser revisitados.

Finalizar Inspeção:

Após completar a inspeção, o operador finaliza a operação do sistema, garantindo o encerramento seguro do drone e o registro das informações coletadas.

5. Diagrama de sequência ou atividades



Fonte: Os autores (2024)

O diagrama de atividades ilustra como o sistema funciona em termos de fluxo de ações, destacando as interações entre o operador e o drone.

5.1 Configuração Inicial

Início do Processo: O operador inicia o sistema através da ação Iniciar Interface, que configura os parâmetros básicos e estabelece a comunicação entre o drone e a estação base.
RNE1: O operador deve estar utilizando uma estação base para controlar o drone, garantindo que todos os comandos e dados sejam transmitidos corretamente.
RNE2: Para que o sistema funcione, o drone deve ser compatível com a estação base, permitindo a conexão entre ambos.

5.2 Verificação da Rota Planejada

O sistema realiza uma verificação automática para determinar se o drone está seguindo a rota planejada no simulador.
Se Sim: O drone continua seguindo o plano, permitindo ao operador marcar pontos de checagem.
Se Não: O operador deve decidir entre duas opções:
Retornar o Drone à Estação Base: Garante a segurança da operação em caso de falha ou problema crítico.
Continuar a Operação: Permite ajustar a rota manualmente ou prosseguir com as ações principais.

5.3 Execução de Ações Principais

Durante a operação, o sistema permite ao operador realizar as seguintes ações de acordo com a necessidade:

Marcar Pontos de Checagem: O operador registra locais específicos de interesse para inspeção, como estruturas que precisam de atenção detalhada.
Movimentar o Drone Manualmente: Caso o drone precise de ajustes na posição ou rota, o operador pode controlá-lo diretamente utilizando a interface do sistema.
Monitorar as Câmeras do Drone: O sistema exibe as imagens captadas pelo drone, permitindo ao operador visualizar as condições das linhas de transmissão.
Essas ações são realizadas com suporte total da estação base, que garante a estabilidade da comunicação e do controle do drone.

5.4 Finalização do Processo

Após a conclusão das tarefas principais (como inspeção e marcação de pontos), o operador pode finalizar a operação retornando o drone à estação base. O fluxo é encerrado com o drone em segurança, pronto para desligamento ou recarga.

5.5 Integração com as Regras de Negócio

As regras de negócio são fundamentais para o funcionamento correto do sistema e estão incorporadas em todo o fluxo:

RNE1 - Estação Base: A utilização de uma estação base é obrigatória, pois ela é o centro de controle de toda a operação, garantindo a comunicação com o drone e o processamento dos dados.
RNE2 - Conexão do Drone: O sistema exige que o drone seja compatível e se conecte à estação base para realizar as tarefas, garantindo a integração e a funcionalidade das ferramentas do sistema.

6. Codificação (código-fonte)

```python
class ROSAdapter:
    def __init__(self):

        rospy.init_node('interface_drone')
        self.vel_pub = rospy.Publisher("/velocidade_manual", Twist, queue_size=10)
        self.mark_pub = rospy.Publisher('/marca', String, queue_size=10)
        self.takeoff_pub = rospy.Publisher('/bebop/takeoff', Empty, queue_size=10)
        self.start_pub = rospy.Publisher("/start", Empty, queue_size=10)
        self.image_sub = rospy.Subscriber("/bebop2/camera_base/image_raw", Image, self.image_callback)
        self.bridge = CvBridge()
        self.image = None

        self.has_takeoff = False

    def move(self, x=0, y=0, z=0, yaw=0):
        velocidade = Twist()
        velocidade.linear.x = x
        velocidade.linear.y = y
        velocidade.linear.z = z
        velocidade.angular.z = yaw
        self.vel_pub.publish(velocidade)

    def stop(self):
        self.vel_pub.publish(Twist())

    def set_mark(self):
        self.mark_pub.publish("marca")

    def start_drone(self):
        msg = Empty()
        if not self.has_takeoff:
            self.takeoff_pub.publish(msg)
            self.has_takeoff = True
        else:
            self.start_pub.publish(msg)

    def image_callback(self, data):
        try:
            cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
            self.image = cv_image
        except CvBridgeError as e:
            print(e)
```

Fonte: Os autores (2024)

A implementação do sistema utiliza uma classe chamada ROSAdapter, que encapsula a comunicação entre a interface do usuário e o drone, usando o framework ROS (Robot Operating System). Essa abordagem segue os princípios da Programação Orientada a Objetos (POO):

- **Encapsulamento**: A classe ROSAdapter agrupa atributos e métodos que gerenciam as operações do drone, como o controle de movimento, marcação de pontos e recepção de imagens da câmera. Essa organização evita o acesso direto a variáveis internas e expõe apenas os métodos necessários para interação com o sistema.

- **Abstração**: A complexidade do ROS e dos tópicos de comunicação é abstraída, permitindo que o operador controle o drone através de métodos intuitivos, como move, start_drone e set_mark, sem precisar entender os detalhes internos do sistema.

- **Polimorfismo e Herança**: Embora a classe ROSAdapter não utilize diretamente herança, ela implementa o conceito de polimorfismo ao se integrar com o ROS. Por exemplo, o método image_callback é projetado para processar mensagens recebidas pelo tópico /bebop2/camera_base/image_raw, permitindo a personalização do processamento de imagens.

Exemplo de métodos principais da classe:

- **move(x, y, z, yaw)**: Publica velocidades no tópico /velocidade_manual para mover o drone.

- **set_mark()**: Marca um ponto de checagem, publicando no tópico /marca.

- **start_drone()**: Gerencia o comando de decolagem ou iniciação de operação, enviando mensagens para tópicos específicos.

- **image_callback(data)**: Processa imagens recebidas da câmera, convertendo-as para o formato OpenCV.

7. Testes

Os testes de caixa-preta consistem em verificar as funcionalidades do sistema sem considerar a lógica ou o código-fonte por trás da interface. A abordagem foca no comportamento do sistema a partir das interações do usuário, avaliando se as saídas estão de acordo com as entradas fornecidas. Abaixo estão os testes planejados:

Funcionalidade	**Cenário de Teste**	**Critério de Sucesso**
Iniciar Interface	O operador inicia a interface e configura os parâmetros iniciais.	A interface é carregada corretamente, sem erros, e conecta o drone à estação base (cumprindo RNE1 e RNE2).
Movimentar Drone Manualmente	Controlar o drone com comandos manuais (mover, subir, descer).	O drone responde a todos os comandos enviados pelo operador.
Marcar Pontos de Checagem	Marcar pontos de checagem no simulador	Os pontos são registrados corretamente no sistema simulado
Monitorar Câmeras do Drone	Visualizar imagens captadas durante a operação.	As imagens capturadas pelo drone são exibidas sem atraso ou interrupções significativas.
Finalizar Operação	Encerrar a operação	O sistema finaliza sem erros.

Fonte: Os autores (2024)

8. Cronograma

Cronograma das tarefas da equipe a serem realizadas até a entrega final do trabalho.

Atividades	**Março**	**Abril**	**Maio**	**Junho**	**Julho**	**Agosto**	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Definição de design da interface	X	X								
Entrega do Pré-Projeto					X					
Defesa Pré-Projeto					X					
Estudar e desenvolver uma interface para o sistema de inspeção que atenda as necessidades e as formas de operação do profissional			X	X	X					
Testes de campo								X	X	
Implementação da visualização/monitoramento do drone por imagens recebidas das câmeras			X	X	X	X	X			
Finalização do primeiro protótipo da interface de controle			X	X						
Apresentação SCiTec									X	
Escrita do artigo final						X	X	X	X	
Entrega do artigo final									X	
Defesa do TCC										X

Fonte: Os autores (2024)