Explorando Algoritmos no Ar: Programação de Drones Estudantis como Ferramenta Didática

Ayslan T. Possebom[1], **Rafael H. D. Zottesso**[1], **Késsia R. C. Marchi**[1], **Daniela E. Flôr**[1]

[1]Instituto Federal do Paraná (IFPR) – Campus Paranavaí/PR
Rua José Felipe Tequinha, 1400, Jardim das Nações – 87703-536 – Paranavaí – Brasil.

ayslan.possebom@ifpr.edu.br, rafael.zottesso@ifpr.edu.br

kessia.marchi@ifpr.edu.br, daniela.flor@ifpr.edu.br

Abstract. *This paper explores the use of programmable drones, particularly the Tello EDU, as a didactic resource for teaching algorithms and data structures. The proposal integrates abstract concepts (sequence, decision, loops, and functions) to practical flight activities, bridging theory and application. Simulation environments with block-based and Python programming were combined with libraries such DroneBlocksTelloSimulator and DJITelloPy for real-world experiments. Expected outcomes include greater student engagement, understanding of both theoretical and practical concepts, and the development of computational thinking.*

Resumo. *Este artigo investiga o uso de drones programáveis, em especial o Tello EDU, como recurso didático para o ensino de algoritmos e estruturas de dados. A proposta integra conceitos abstratos da disciplina (sequência, decisão, repetição e funções) a atividades práticas de voo, aproximando teoria e aplicação. Foram utilizados ambientes de simulação em blocos e em Python, além de bibliotecas como DroneBlocksTelloSimulator e DJITelloPy para experimentações reais. Os resultados esperados incluem maior engajamento e, compreensão de conceitos teóricos e práticos, e desenvolvimento de pensamento computacional.*

1. Introdução

Ensinar algoritmos e estruturas de dados representa um dos maiores desafios enfrentados no ensino de Computação, tanto para jovens em formação inicial quanto para adultos em processos de requalificação profissional. Embora o conteúdo seja essencial para o desenvolvimento do pensamento computacional, a abstração envolvida nos conceitos de variáveis, estruturas de decisão e repetição, e modularização de códigos frequentemente dificultam sua compreensão inicial, mediante a dificuldade em resolver problemas cuja solução depende da construção progressiva ao longo do tempo para se obter um resultado final. Estudantes tendem a enxergar algoritmos como construções puramente teóricas, desprovidas de aplicação tangível, o que resulta em desmotivação e dificuldade de aprendizado.

Nesse contexto, torna-se fundamental a adoção de abordagens práticas e lúdicas, capazes de transformar conceitos abstratos em experiências concretas

[Ng and Cheng 2019]. A experimentação prática permite compreender os passos de execução e explorar a lógica de forma interativa, ampliando engajamento e internalização dos conceitos [Sattar and Nawaz 2023].

Entre as possibilidades tecnológicas, o uso de drones educacionais e IoD (*Internet of Drones*) tem emergido como uma alternativa promissora [Pergantis and Drigas 2024]. Além do apelo motivacional natural associado a dispositivos robóticos e voadores, drones oferecem um ambiente rico para a experimentação de algoritmos em situações do mundo real, permitindo que estudantes observem a execução de estruturas de dados e fluxos de controle por meio de movimentos físicos e interações com o espaço. O drone Tello EDU, programável em linguagem como Python, destaca-se por sua acessibilidade, robustez e adequação ao ambiente educacional [Gasmi and Cherfaoui 2023], sendo uma ferramenta ideal para integrar teoria e prática.

Este trabalho tem como objetivo geral investigar o uso de drones programáveis como recurso pedagógico no ensino de algoritmos e estruturas de dados, buscando responder à seguinte pergunta de pesquisa: de que forma a programação de drones pode auxiliar na compreensão e aplicação prática de conceitos fundamentais de algoritmos e estruturas de dados? Como objetivos específicos, propõe-se: (i) mapear conteúdos clássicos da disciplina de Algoritmos e Estruturas de Dados a desafios de programação com drones; e (ii) desenvolver e aplicar exemplos práticos utilizando Python e o drone Tello EDU físico e simulado.

O artigo está organizado da seguinte forma: na Seção 2, apresenta os trabalhos relacionados sobre o uso de tecnologias educacionais no ensino de algoritmos e pensamento computacional. Na Seção 3, é descrito a metodologia empregada e os desafios propostos utilizando o drone Tello EDU. A Seção 4 discute os resultados e as análises obtidas a partir das experiências práticas utilizando o drone físico Tello EDU. Finalmente, a Seção 5 apresenta as conclusões, limitações e direções para trabalhos futuros.

2. Trabalhos relacionados

A literatura recente sobre o uso de drones na educação aponta ganhos motivacionais e cognitivos. Os modelos de drones mais populares usados na educação, em especial, na codificação, tais como ROBOlink CoDrone Edu, Tello Edu, Parrot MAMBO Educational, Crazyflie 2.1 (bitcraze), ou plataformas de controle de voo como ArduPilot e PX4, permitem aplicar linguagens de programação (ex: Python) para o controle do drone. Desta forma, pode-se desenvolver algoritmos para tais dispositivos, usando instruções para o controle de fluxo, tomada de decisões, repetições, modularização de códigos e estruturas de dados.

No trabalho desenvolvido por [Bai et al. 2021], desenvolveu-se uma revisão crítica das plataformas educacionais (incluindo Tello EDU, CoDrone, Sky Viper, Parrot Mambo), mapeando usos do ensino fundamental ao ensino superior. Em matemática e física, o trabalho enfatiza possibilidades de aplicações de drones em modelagem matemática, aerodinâmica, entradas de controle e fotogrametria, além de exemplos curriculares que envolvem cálculo de velocidade e distância (matemática) e massa/peso (física). Os autores apresentam o forte apelo motivacional aos estudantes como principal vantagem do uso de drones. Em contrapartida, existe a necessidade de organização curricular para assegurar profundidade conceitual a ser aplicado em cada atividade.

O trabalho apresentado por [Sattar and Nawaz 2023] desenvolve uma abordagem pedagógica com Tello EDU e programação em blocos (DroneBlocks) em seis escolas australianas, na qual os estudantes automatizam voos em padrões geométricos (reta, arco, retângulo, triângulos, zigue-zague) e manobras, articulando algoritmos matemáticos para programar sequências e assim estimular pensamento computacional (formulação, decomposição). Os autores afirmam que houveram ganhos significativos em pensamento computacional e resolução de problemas, entretanto, o estudo aborda apenas linguagem visual (blocos).

O trabalho apresentado por [Ng and Cheng 2019] desenvolve um estado da arte do Tello EDU em projetos educacionais, além de mapear aplicações educacionais (missões básicas, enxames/*swarm*, IA) e projetos (ex: obstáculos, busca e salvamento). Os conceitos trabalhados incluem programação e práticas de engenharia, entretanto, o foco do trabalho é predominantemente técnico (implementação e controle), com menos ênfase em atividades pedagógica básicas para ensino de conceitos, lógica e linguagem de programação.

Diferentemente de abordagens majoritariamente baseadas em blocos, este artigo foca no desenvolvimento de raciocínio lógico e introdução ao pensamento computacional, em especial, utilizando a linguagem de programação Python com o drone Tello Edu. Os conteúdos pedagógicos envolvidos visam tornar as disciplinas de Algoritmos, Linguagem de Programação e Estruturas de dados mais dinâmicas, promovendo debates sobre instruções, lógica e competições.

3. Programação de Drones

O Tello EDU é um minidrone educacional desenvolvido pela Ryze Technology em colaboração com DJI e Intel, pensado especificamente para o ensino de programação, robótica e conceitos STEM (Science, Technology, Engineering, and Mathematics). Ele vem preparado para programação via SDK/API, oferecendo controle de voo em diferentes níveis e acesso ao *stream* de vídeo, o que facilita a integração com atividades de lógica, algoritmos e visão computacional. Para atender perfis distintos de alunos e disciplinas, estão disponíveis múltiplos ambientes e linguagens (ex: Python, Scratch) permitindo desde sequências de blocos para iniciantes até projetos textuais mais avançados que exploram estruturas de dados e modularização de código.

O Tello Edu é um drone compacto e leve, possuindo câmera integrada, sistema de posicionamento visual, barômetro e conectividade Wi-Fi, permitindo o desenvolvimento de aplicações para IoD. Seu uso é indicado para ambientes internos, com tempo de voo em torno de 13 minutos, alcance de até 100 m, velocidade máxima próxima de 8 m/s e altura de voo de cerca de 30 m. Esses recursos tornam o drone um bom suporte para atividades que envolvem geometria de trajetórias, medição de deslocamento/velocidade (matemática/física) e coleta/análise de imagens (computação).

A Figura 1 apresenta o drone Tello Edu. Ele é composto por 4 motores. Como um quadricóptero, dois motores giram em CW (*clockwise* - sentido horário) e dois em CCW (*counter-clockwise* - sentido anti-horário), em pares diagonais. Isso serve para anular o torque (não girar sozinho) e permitir o *yaw* (girar no próprio eixo) acelerando um par de motores enquanto desacelera o outro par de motores.

Ao ligar o drone, ele inicializa uma rede Wifi (geralmente chamada de Tello seguido pelo modelo do drone) sem senha. Um telefone celular, *tablet* ou computador pode

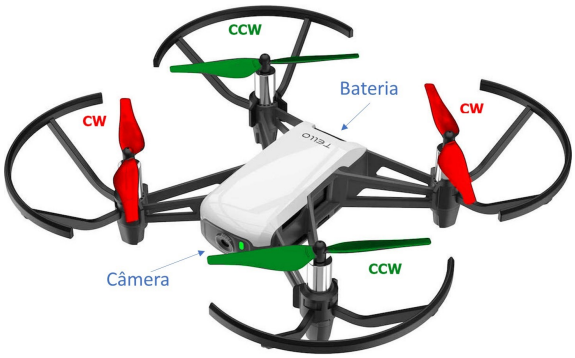

Figura 1. Drone Tello Edu.

ser utilizado para se conectar ao drone utilizando a rede Wifi. O site da DJI oferece o aplicativo Tello para download em iOS e Android [1]. Após conectar na rede wifi com um telefone celular, o aplicativo Tello reconhecerá o drone e fornecerá o controle do drone, conforme apresentado na Figura 2.

Figura 2. App Tello: controle remoto do drone.

3.1. Ambiente de Simulação: Programação em Blocos

A plataforma DroneBlocks Simulator[2] possui um ambiente visual que pode ser utilizado para o ensino de pensamento computacional e algoritmos por meio de programação em blocos aplicada a veículos aéreos não tripulados (VANTs). O simulador se propõe a replicar as condições físicas de um voo do drone DJI Tello Edu, permitindo aos estudantes projetarem e testarem diferentes missões de forma acessível, sem necessidade de acesso ao hardware real.

A programação em blocos facilita a compreensão de conceitos fundamentais de lógica, como sequenciamento, uso de estruturas condicionais e laços de repetição, promovendo a aplicação prática de conhecimentos matemáticos e computacionais em cenários

[1]Disponível em: `https://www.dji.com/br/downloads/djiapp/tello`. Acessado em 15/09/2025.

[2]Disponível em: `https://dev.droneblocks.io/simulator.html`. Acessado em: 17/09/2025.

reais de navegação autônoma. Além disso, o ambiente de simulação permite o acompanhamento visual das trajetórias do drone e fornece feedback imediato sobre a execução dos algoritmos criados pelos estudantes.

O DroneBlocks Simulator funciona de forma online, em sistema baseado na web, acessado por um navegador web. A plataforma viabiliza o envolvimento ativo dos estudantes, favorecendo a experimentação de instruções de forma visual e contribuindo como ferramenta pedagógica ao ensino de programação de sistemas. No simulador, diversos ambientes estão disponíveis, simulando o voo do drone em diferentes condições, tais como ambiente minimalista (pista de decolagem e pouso), ambiente de inverno e neve, cidade, exploração de outros planetas, montanhas, exploração do Egito, entre outros.

Ao acessar a plataforma e lançar o ambiente desejado, é solicitado uma senha (*passcode*). Este código é fornecido após realizar o cadastro no sistema e se inscrever em algum curso gratuito. A Figura 3 apresenta o ambiente Tundra, representando um ambiente de inverso e neve.

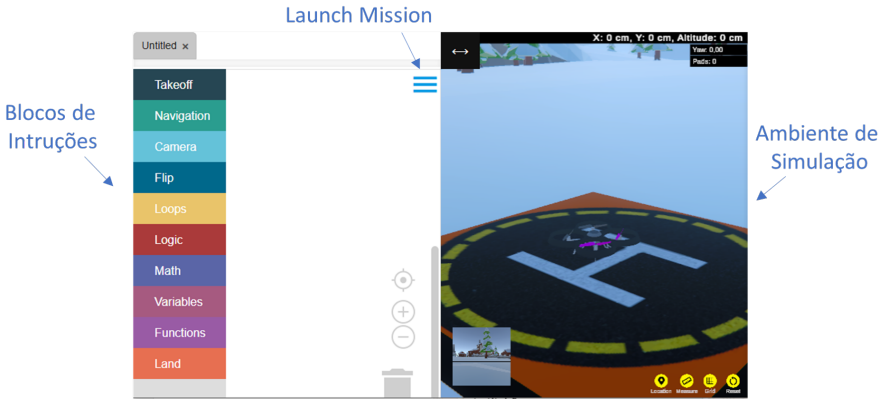

Figura 3. Ambiente de desenvolvimento em blocos do DroneBlocks Simulator.

Entre os blocos de instruções, tem-se as opções:

- *Takeoff*: instruções para decolagem do drone;
- *Navigation*: instruções para estabelecer limites de velocidades, voar para diferentes direções (frente, trás, esquerda, direita, cima, baixo ou posição específica do cenário), fazer curvas, pairar e girar no sentido horário;
- *Camera*: instruções para tirar fotos com a câmera do drone;
- *Flip*: instruções para realizar manobras de giro (girar para esquerda, direita, frente e trás);
- *Loops*: instruções para controle de repetições, tais como repetições contadas ou controladas por condição;
- *Logic*: instruções para tomada de decisão e operadores relacionais/lógicos;
- *Math*: instruções para definição de valores, cálculos matemáticos, raiz quadrada, seno, valor de π, arredondamento de valores, números aleatórios, entre outras instruções matemáticas;
- *Variables*: permite definir variáveis para serem utilizadas no algoritmo;
- *Functions*: definir blocos de instruções com ou sem retorno de valores;
- *Land*: instruções para pousar o drone.

A Figura 4 apresenta um exemplo de algoritmo desenvolvido em blocos. Cada bloco (instrução) representa uma ação do drone. O exemplo possui um conjunto de instruções sequencial, onde o drone faz a decolagem, voa para frente, voa para a esquerda, voa para trás, voa para a direita, voa novamente para trás e, por fim, pousa.

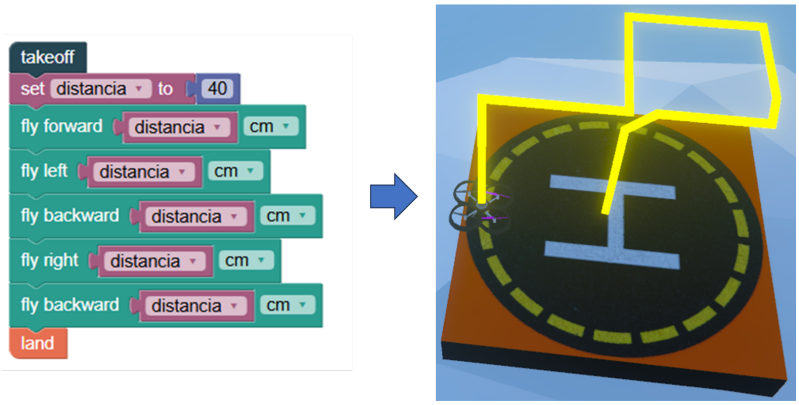

Figura 4. Instruções em Bloco e Execução do Simulador.

3.2. Ambiente de Simulação Online: Programação em Python

Além da simulação dos voos implementada por programação em blocos, também é possível utilizar a linguagem de programação Python para desenvolver os algoritmos a serem executados pelo drone utilizando o DroneBlocks Simulator. Com esta abordagem, é possível interagir com um drone Tello em ambiente simulado e utilizar o mesmo código para execução no drone Tello Edu real [Baldwin 2024].

Para a instalação desta biblioteca, utiliza-se o comando `pip install DroneBlocksTelloSimulator` no Prompt de Comandos. Para a simulação, o navegador precisa estar com a opção de "Permitir conteúdo inseguro" habilitada, caso contrário, não será permitido a execução do simulador no navegador web. O simulador web está disponível em `https://coding-sim.droneblocks.io/`. Ao abrir o simulador, deve-se clicar no botão *Get Drone Simulator Key*. Esta chave exibida na tela será utilizada no código em Python para que o computador possa se comunicar com o simulador e visualizar a execução do algoritmo na tela do navegador web.

Algumas ações possíveis no simulador são:

- **drone.takeoff()**: decolar;
- **drone.land()**: pousar;
- **drone.set_speed(velocidade)**: controla a velocidade do drone (valores $0 \sim 100$);
- **drone.fly_forward(distancia, medida)**, **drone.fly_left(distancia, medida)**, **drone.fly_right(distancia, medida)** e **drone.fly_backward(distancia, medida)**: faz o drone voar para frente, esquerda, direita e para tráz, respectivamente, em uma distância informada na unidade de medida 'cm' ou 'in';
- **drone.fly_up(distancia, medida)** e **drone.fly_down(distancia, medida)**: faz o drone subir e descer, respectivamente;
- **drone.flip_left()**, **drone.flip_right()**, **drone.flip_forward()** e **drone.flip_backward()**: faz o drone rotacionar em uma manobra para a esquerda, direita, para frente e para trás, respectivamente;

- **drone.yaw_right(graus)** e **drone.yaw_left(graus)**: faz o drone rotacionar para a direita ou esquerda, respectivamente, no número de graus informado (valores 1 ∼ 360);
- **drone.fly_to_xyz(x, y, z, unidades)**: leva o drone a uma posição (x, y, z) no espaço 3D do simulador, em linha reta, com valores representados pelas unidades 'in' ou 'cm';
- **drone.fly_curve(x1, y1, z1, x2, y2, z2, unidades)**: faz o drone percorrer um arco que sai da posição atual, passa pelo ponto intermediário (x1,y1,z1) e termina no ponto (x2,y2,z2), nas distâncias indicadas por 'in' ou 'cm'.

Tendo-se as funções básicas para controle do drone, pode-se implementar o algoritmo em Python. Um exemplo de implementação é apresentado na Listagem 1, substituindo o valor da variável `sim_key` para o código fornecido no site do simulador.

Listagem 1. Código em Python do controle básico do DroneBlocks Simulator

```
1  from DroneBlocksTelloSimulator.DroneBlocksSimulatorContextManager import
       DroneBlocksSimulatorContextManager
2
3  sim_key = 'drone-simulator-key'
4
5  distancia = 50
6
7  with DroneBlocksSimulatorContextManager(simulator_key=sim_key) as drone:
8     drone.takeoff()
9     drone.fly_forward(distancia, 'cm')
10    drone.yaw_left(90)
11    drone.fly_forward(distancia, 'cm')
12    drone.fly_right(distancia, 'cm')
13    drone.flip_right()
14    drone.yaw_right(180)
15    drone.fly_backward(distancia, 'cm')
16    drone.land()
```

Ao executar o código, o algoritmo se conectará ao sistema web do simulador e será possível visualizar o drone voando, conforme a sequência de instruções implementadas. A Figura 5 representa o trajeto percorrido pelo drone. Os números correspondem às linhas de cada instrução na listagem.

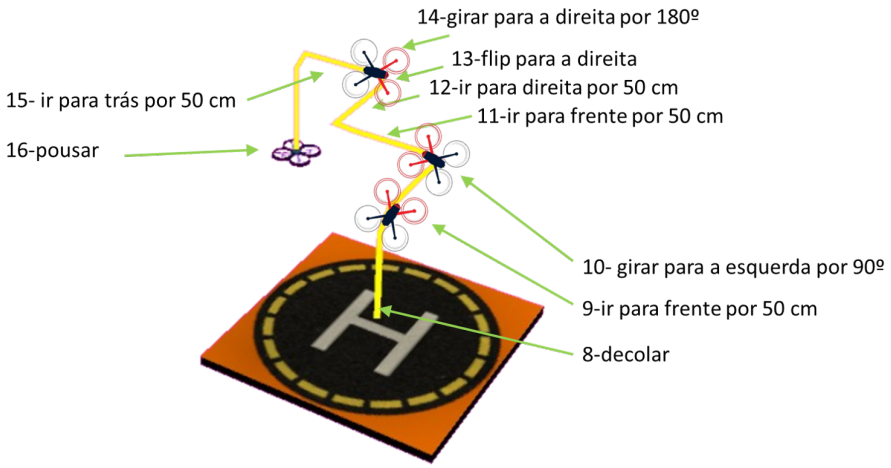

Figura 5. Simulação. Sequência de instruções da Listagem 1 (linhas).

3.3. Bibliotecas em Python para voos reais

As bibliotecas **DJITelloPy** [Escoté 2023], **TelloPy** [Hanyazou 2018] e **EasyTello** [Ezra Fielding 2019] apresentam-se como as opções mais populares para programar drones Tello EDU em Python, cada uma com um nível de abstração distinto e implicações didático-pedagógicas diferentes:

- DJITelloPy: biblioteca "pronta para uso" que abstrai o SDK/UDP da Tello e oferece métodos diretos para decolagem/pouso, movimentos em cm, rotações, *flips* e *stream* de vídeo, facilitando a integração com OpenCV. É adequada para trabalhar com laços, condicionais, funções e modularização, além de aplicações de visão computacional/*Machine Learning*;
- TelloPy: Fornece controle do drone com foco em telemetria (posição, velocidade, altura, bateria, temperatura, aceleração, altitude) e eventos (término de comandos, perda de conexão, atualizações de vídeo), permitindo monitoramento em tempo real e reação assíncrona ao estado do voo;
- EasyTello: Implementa os comandos básicos do SDK (decolagem, movimentos, rotações, *flips*), ideal para introdução rápida. Em contrapartida, tem recursos avançados limitados (telemetria, *stream*, etc.), sendo indicada para aulas práticas iniciais (Internet dos Drones).

A Tabela 1 sintetiza instalação, importação, criação do objeto e conexão com o drone. A Tabela 2 compara funções de movimentação. A Tabela 3 contrasta velocidade, *flip*, transmissão de imagens, barômetro e leitura de bateria entre as diferentes bibliotecas.

Tabela 1. Bibliotecas: importação, criação do objeto e conexão.

Biblioteca	DJITelloPy	TelloPy	EasyTello
Site	https://github.com/damia-fuentes/DJITelloPy	https://github.com/hanyazou /TelloPy	https://github.com/ezrafielding/easyTello
Instalação	`pip install djitellopy`	`pip install tellopy`	`pip install easytello`
Importação de biblioteca	`from djitellopy import Tello`	`import tellopy`	`from easytello import tello`
Objeto do Tello (drone)	`drone = Tello()`	`drone = tellopy.Tello()`	`drone = tello.Tello()`

Conectar ao drone	`drone.connect()`	`drone.connect()`
		`drone.wait_for_connection(60)`
		Tempo para conexão: 10 a 60
			Ao criar o objeto, a conexão é estabelecida automaticamente.

Após o desenvolvimento do algoritmo, basta conectar o computador na mesma rede WiFI específica do drone Tello Edu e executar o código Python. Após a fase de tradução do código, ele é enviado pela rede para o dispositivo físico do drone e, em seguida, o voo é inicializado.

4. Experiências práticas

Nesta seção, foi adotada a DJITelloPy como base, por ser amplamente utilizada na comunidade do Tello/Tello EDU, facilitando a construção de atividades que apliquem conhecimentos introdutórios aos estudantes de algoritmos e desenvolvimento de pensamento computacional.

4.1. Instruções sequenciais

As instruções sequenciais representam a base da programação, permitindo a execução linear de comandos para criar soluções previsíveis e determinísticas. No contexto do

Tabela 2. Bibliotecas: movimentação de drones.

Biblioteca	DJITelloPy	TelloPy	EasyTello
Decolar	`drone.takeoff()`	`drone.takeoff()`	`drone.takeoff()`
Aterrissar	`drone.land()`	`drone.land()`	`drone.land()`
Mover Esquerda	`drone.move_left(100)` Valores: 20–500 cm	`drone.left(50)` Valores: 0–100	`drone.left(100)` Valores: 20–500 cm
Mover Direita	`drone.move_right(100)` Valores: 20–500 cm	`drone.right(50)` Valores: 0–100	`drone.right(100)` Valores: 20–500 cm
Girar sentido horário	`drone.rotate_clockwise(100)` Valores: 1–360°	`drone.clockwise(50)` Valores: 0–100	`drone.cw(50)` Valores: 1–360°
Girar sentido anti-horário	`drone.rotate_counter_clockwise(90)` Valores: 1–360°	`drone.counter_clockwise(50)` Valores: 0–100	`drone.ccw(50)` Valores: 1–360°
Mover para frente	`drone.move_forward(100)` Valores: 20–500 cm	`drone.forward(50)` Valores: 0–100	`drone.forward(100)` Valores: 20–500 cm
Mover para trás	`drone.move_back(100)` Valores: 20–500 cm	`drone.backward(50)` Valores: 0–100	`drone.back(100)` Valores: 20–500 cm
Mover para baixo	`drone.move_down(100)` Valores: 20–500 cm	`drone.down(50)` Valores: 0–100	`drone.down(100)` Valores: 20–500 cm
Mover para cima	`drone.move_up(100)` Valores: 20–500 cm	`drone.up(50)` Valores: 0–100	`drone.up(100)` Valores: 20–500 cm

Tabela 3. Bibliotecas: *streaming*, *flips*, barômetro e velocidade.

Biblioteca	DJITelloPy	TelloPy	EasyTello
Velocidade	`drone.set_speed(50)` Valores: 10–100 cm	Sem funções pré-definidas. Apenas com registro de eventos.	`drone.set_speed(50)` Valores: 10–100 cm

Flip	`drone.flip_back()`
	`drone.flip_forward()`
	`drone.flip_left()`
	`drone.flip_right()`
		`drone.flip_back()`
		`drone.flip_forward()`
		`drone.flip_left()`
		`drone.flip_right()`
			`drone.flip(direcao)`
			”l”left
			”r”right
			”f”forward
			”b”back

Iniciar transmissão de imagem	`drone.streamon()` `drone.get_frame_read()`	`drone.start_video()` `drone.get_video_stream()`	`drone.streamon()`
Finalizar transmissão de imagem	`drone.streamoff()`	`drone.quit()`	`drone.streamoff()`
Obter porcentagem de bateria restante	`drone.get_battery()` Valores: 0 a 100%	Sem funções pré-definidas. Apenas com registro de eventos.	`drone.get_battery()` Valores: 0 a 100%

Obter altitude (barômetro)	`drone.get_barometer()`
	`drone.get_height()`
	Valor: em cm
		Sem funções pré-definidas. Apenas com registro de eventos.	`drone.get_height()`
			`drone.get_baro()`

drone Tello Edu, elas são ideais para implementar trajetórias pré-definidas, onde cada ação é executada em ordem, promovendo o entendimento de fluxo de controle básico e a visualização imediata dos resultados no comportamento do drone. Alguns exemplos de atividades:

- Realizar uma trajetória quadrada simples, com cada lado do quadrado medindo 50cm;
- Realizar uma trajetória circular simples, com diâmetro definido pelo estudante;
- Fazer com que o drone decole de um ponto específico e pouse em outro ponto definido (pista);
- Realizar uma trajetória aleatória (frente, trás, direita, esquerda, com rotações);
- Programar o drone para executar uma sequência de subidas e descidas alternadas, simulando um padrão de onda;
- Criar uma rotina de inspeção simples, onde o drone se move em zigue-zague para cobrir uma área retangular pré-determinada;
- Desenvolver uma sequência de comandos para o drone realizar uma volta completa de 360 graus no ar, retornando à posição inicial.

4.2. Estruturas de decisão/condicionais

As estruturas de decisão ou condicionais introduzem a capacidade de bifurcação no fluxo do programa, permitindo que o drone responda dinamicamente a condições ambientais ou internas. Pode-se utilizar variáveis para armazenar valores aleatórios ou realizar cálculos matemáticos, em especial, ao se obter dados do drone (ex.: altura, bateria, etc.), fomentando o raciocínio lógico e a adaptação em tempo real. Alguns exemplos de atividades:

- Dependendo do nível da bateria, o drone deverá executar algum trajeto ou pousar;
- Executar caminhos por uma quantia de tempo;
- Escolha de caminhos com base na altura do drone;
- Verificar a temperatura do drone e, se acima de um limite, reduzir a velocidade dos movimentos para evitar superaquecimento;
- Decidir entre rotas alternativas com base na detecção de um marcador simples (*mission pad*) via sensor do drone;
- Avaliar o tempo de voo acumulado e, se ultrapassar um limiar, executar uma manobra de retorno à base.

4.3. Estruturas de repetição

As estruturas de repetição, como os laços *for* e *while*, são essenciais para automatizar tarefas iterativas, otimizando o código e permitindo a execução de padrões complexos com eficiência. Para a aplicação de estruturas de repetição (instruções *for* e *while*), diversos tipos de problemas podem ser propostos, tais como aqueles que envolvem acumulação de valores ou repetições controladas por condições, incentivando o estudante a pensar em eficiência e terminação de *loops*. Alguns exemplos de atividades:

- Percorrer uma trajetória de quadrado com quatro lados, de forma que ao pousar, o drone deve estar na mesma posição e direção do momento da decolagem;
- Percorrer uma trajetória em linha reta por uma distância específica, de forma que o drone voe acumulando distâncias (soma de valores) a cada repetição, até atingir a distância especificada;
- Programar o drone para realizar um padrão de voo em forma de estrela, repetindo movimentos em um laço. Por exemplo, realizar cinco movimentos retos de 50 cm, girando 144 graus após cada movimento;
- Criar um *loop* para o drone girar em círculos concêntricos, aumentando o raio a cada iteração até atingir um limite definido;
- Implementar um padrão de busca em grade, onde o drone repete movimentos em linhas paralelas para mapear uma área;
- Usar um laço *while* para manter o drone pairando até que a bateria atinja um nível mínimo, ajustando altitude periodicamente.

4.4. Funções e parâmetros

As funções com parâmetros e retorno de valores promovem a modularidade e reutilização de código, permitindo que os estudantes organizem soluções complexas em blocos independentes e testáveis. A aplicação dos conceitos de modularização de códigos permite a criação de bibliotecas personalizadas para manobras reutilizáveis, incentivando práticas de programação estruturada. Alguns exemplos de funções:

- Função que permita ao drone realizar um movimento em espiral, com parâmetros para definir o raio e a altura do movimento, retornando o status de conclusão;
- Função para calcular e executar uma trajetória reta, recebendo distância e direção como parâmetros, e retornando o tempo estimado de execução;
- Função de verificação de ambiente, que recebe dados de sensores (como altura) e retorna uma decisão booleana para prosseguir ou abortar a missão;
- Função para gerar padrões repetitivos, como quadrados ou círculos, com parâmetros para o número de repetições e tamanho, retornando o consumo de bateria estimado.

4.5. Visão computacional

A câmera integrada do drone Tello Edu é uma ferramenta poderosa para introduzir conceitos de visão computacional de forma acessível, permitindo que os estudantes capturem imagens e processem dados visuais em tempo real. Utilizando a biblioteca DJITelloPy, é possível acessar o fluxo de vídeo do drone e realizar tarefas como tirar fotos ou detectar objetos com o auxílio da biblioteca **OpenCV**. Essas atividades são particularmente úteis para ensinar processamento de imagens e integração de dados visuais com comandos de controle do drone, promovendo um aprendizado interdisciplinar que combina programação, algoritmos e robótica.

A Listagem 2 demonstra como capturar uma foto com a câmera do Tello Edu. Este código é simples e didático, ideal para iniciantes, pois introduz o controle básico do fluxo de vídeo e o salvamento de imagens. O drone inicia o streaming de vídeo, captura um frame e o salva como um arquivo de imagem.

Listagem 2. Capturando imagens com DJITelloPy

```
from djitellopy import Tello
import cv2
import time

drone = Tello() # Inicializa o drone
drone.connect()
drone.streamon() # Inicia o fluxo de video
time.sleep(2) # Aguarda a estabilizacao do streaming
frame_read = drone.get_frame_read() # Captura um frame do video
frame = frame_read.frame
cv2.imwrite("foto_drone.jpg", frame) # Salva a imagem capturada
drone.streamoff() # Finaliza o streaming
```

Dependendo do conhecimento técnico em OpenCV, algoritmos mais aprimorados poderão ser desenvolvidos, tal como detectar um papel de uma cor específica e fazer com que o drone siga sua direção, ajustando o movimento com base na posição do objeto.

5. Conclusões

Este artigo apresentou uma proposta de uso do Tello EDU programado em Python para aproximar conteúdos de Algoritmos e Estruturas de Dados de experiências práticas em sala de aula. Foram mapeados conceitos centrais (sequência, decisão, repetição e modularização) a atividades de voo progressivas e replicáveis, incluindo exemplos de códigos introdutórios a serem desenvolvidos em ambiente simulado e ambiente real. A contribuição reside em transformar abstrações tradicionais (variáveis, acumuladores, laços e funções) em comportamentos observáveis, reduzindo a distância entre o raciocínio algorítmico e seus efeitos no mundo físico.

Do ponto de vista pedagógico, os resultados esperados incluem maior engajamento, feedback imediato sobre a lógica implementada e desenvolvimento do pensamento computacional. A estratégia favorece a interdisciplinaridade (matemática e física básica para trajetórias, ângulos, velocidades e alturas) e incentiva boas práticas de engenharia de software (parametrização, contratos simples de funções, registro de métricas).

Foram identificadas, entretanto, limitações importantes a considerar: tempo de voo reduzido e dependente de bateria, pouca variabilidade de sensores (altura/barômetro) e latência de rede Wi-Fi.

Como trabalhos futuros, propõe-se avaliar empiricamente a eficácia da proposta em turmas reais, considerando o engajamento dos estudantes e motivação quanto ao aprendizado e aplicação prática em dispositivos robóticos. Pretende-se também estender os desafios para tópicos mais avançados (pilhas/filas, grafos com missões) e investigar estratégias de formação docente que acelerem a integração curricular.

Referências

Bai, O., Chu, H., Liu, H., and Hui, G. (2021). Drones in education: A critical review. *Turkish Journal of Computer and Mathematics Education*, 12(11):1722–1727.

Baldwin, D. (2024). Droneblockstellosimulator. `https://pypi.org/project/DroneBlocksTelloSimulator/`. Version 0.0.8. PyPI package. MIT License. Maintainer: DroneBlocks.

Escoté, D. F. (2023). Djitellopy: Dji tello drone python interface using the official tello sdk. `https://github.com/damiafuentes/DJITelloPy`. Acessado em 23/09/2025.

Ezra Fielding, V. K. (2019). easytello: An easy to use library to support dji tello scripting in python 3. `https://github.com/ezrafielding/easyTello`. Acessado em 23/09/2025.

Gasmi, K. and Cherfaoui, Y. (2023). Using the tello edu drone for educational purposes. Master thesis in telecommunication systems and networking, Ecole Nationale Supérieure des Technologies Avancées (ENSTA), Algiers, Algeria. Department: Génie Électrique et Informatique Industrielle (GEII).

Hanyazou (2018). Tellopy: Dji tello drone controller python package. `https://github.com/hanyazou/TelloPy`. Acessado em 23/09/2025.

Ng, W. S. and Cheng, G. (2019). Integrating drone technology in stem education: A case study to assess teachers'readiness and training needs. *Issues in Informing Science & Information Technology*, 16.

Pergantis, P. and Drigas, A. (2024). The effect of drones in the educational process: A systematic review. *Education Sciences*, 14(6):665.

Sattar, F. and Nawaz, M. (2023). Developing computational thinking in stem education with drones. In *2023 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–5. IEEE.