Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

SimuDrones: A Browser-Based 3D Virtual Laboratory for Drone Design, Rule-Based Hardware Validation, and Physics-Simulated Flight

Jayesh Ambekar

jayeshambekar31@gmail.com

SNJB's KBJ College of Engineering

Pranjal Batiya

SNJB's KBJ College of Engineering

Suchita Jadhav

SNJB's KBJ College of Engineering

Research Article

Keywords:

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10042500/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

SimuDrones: A Browser-Based 3D Virtual Laboratory for Drone Design, Rule-Based Hardware Validation, and Physics-Simulated Flight

Jayesh Ambekar[1], Pranjal Batiya[1], Suchita Jadhav[1]

[1] Department of Computer Engineering, SNJB's KBJ College of Engineering, Nashik, Maharashtra, India .

Contributing authors: jayeshambekar31@gmail.com; pranjalbatiya87@gmail.com; suchita19.c@gmail.com;

Abstract

Purpose: Existing drone simulation environments address flight physics or component configuration in isolation and typically demand complex local installation, dedicated hardware, and expert-level setup. No unified, browser-accessible platform covers the complete engineering workflow from part selection and three-dimensional assembly to electrical wiring and physics-based test flight. This work presents SimuDrones, which closes that gap for educational settings.

Methods: SimuDrones is structured in four layers: (i) a structured component database encoding real-world electrical and mechanical specifications for motors, ESCs, batteries, propellers, and flight controllers; (ii) a WebGL-based 3D assembly environment built on React Three Fiber and Zustand; (iii) a sequential 19-rule hardware-compatibility validation engine; and (iv) a real-time Euler-integration physics flight simulator that derives all force parameters directly from the user-assembled drone specification. An interactive node-graph circuit builder covers three professional flight controllers. The system is implemented as a Next.js 16 web application requiring zero local installation.

Results: Controlled evaluation across 47 curated build configurations— covering correct builds, single-fault injections across all 19 rule categories, and compound multi-fault cases—achieved 100% precision and recall. The Euler-integration hover-throttle model deviated by at most $\pm 4.2\%$ from the Newtonian analytical ground truth across five drone weight classes (200 g to 1400 g), with propeller spin-up/spin-down dynamics yielding $\boldsymbol{R^2 = 0.97}$. Application load time averaged 2.8 s on broadband hardware at a stable 60 fps render rate.

Conclusion: SimuDrones constitutes the first unified, zero-installation browser platform integrating 3D drone assembly, automated hardware validation, interactive flight-controller wiring, and physics-based flight simulation, offering an engineering-grade educational tool for university drone curricula and online learning environments.

Keywords: Drone simulation, WebGL, Hardware validation, Flight controller, Browser-based education, React Three Fiber, Quadrotor

1 Introduction

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have transitioned from niche research instruments to mainstream tools employed across agriculture, infrastructure inspection, disaster response, aerial photography, and logistics [1, 2]. As commercial and regulatory interest in UAV technology intensifies—particularly under India's Civil Aviation Requirements UAS Rules 2021 and the DGCA DigitalSky framework [3]—universities and online learning platforms face mounting demand for structured, low-cost drone engineering curricula.

Physical drone education carries significant barriers. A typical FPV racing or survey quadrotor involves selecting and matching frames, motors, electronic speed controllers (ESCs), batteries, flight controllers (FCs), propellers, and receiver hardware at a combined cost of $200–$800. Incorrect component pairings— mismatched voltage ratings, insufficient ESC current capacity, unbalanced center of gravity—lead to hardware damage or flight incidents. Conventional simulation tools address isolated aspects of this problem. Desktop simulators such as Gazebo/ROS and Microsoft AirSim [4, 5] offer physics-accurate flight dynamics but require complex local installation and provide no component selection, electrical wiring, or hardware pre-validation. Circuit diagram tools such as easyEDA and DroneBlocks facilitate schematic design but lack 3D spatial assembly and physics. No existing platform unifies the complete engineering workflow—from part selection and 3D assembly through wiring to test flight—in a single zero-install browser environment.

This paper presents **SimuDrones**, a browser-based 3D virtual drone laboratory that bridges this gap. The principal contributions of this work are:

1. A **structured real-world component database** encoding electrical and mechanical specifications for all major quadrotor subsystems, sourced from manufacturer datasheets.
2. A **WebGL-based interactive 3D assembly environment** with drag-and-drop GLB part placement, snap-to-socket positioning, 20-step temporal undo, and an exploded-view inspection mode.
3. A **19-rule sequential hardware-compatibility validation engine** that detects electrical, mechanical, and geometric incompatibilities at design time, with 100% precision and recall over 47 tested build configurations.

4. An **interactive node-graph circuit wiring module** for three professional flight controllers— Pixhawk 2.4.8, CUAV V5+, and Kakute F7 AIO—enforcing 25 mandatory connection pairs and rejecting all invalid port-to-port pairings.
5. A **Euler-integration physics flight simulator** whose force parameters are derived entirely from the user-assembled drone specification, producing hover-throttle estimates within ±4.2% of Newtonian analytical values.
6. A **Drone Knowledge Hub** providing drone categorisation, component reference material, and India-specific DGCA regulatory guidance across four UAS weight classes.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 describes the overall system architecture. Sections **??**–8 detail the five platform modules. Section 9 presents evaluation results. Section 10 discusses limitations and future directions. Section 11 concludes.

2 Related Work

2.1 Desktop UAV Simulators

Gazebo, integrated with the Robot Operating System (ROS), is the most widely used open-source multi-robot simulator for UAV research [4]. It supports plugin-based sensor modelling, accurate aerodynamic forces, and integration with flight stacks such as PX4 [6] and ArduPilot. However, Gazebo requires a Linux environment, ROS installation, and substantial configuration effort, making it unsuitable for introductory engineering courses or self-directed learners. Microsoft AirSim [5] provides photorealistic visual environments and is tightly coupled with the Unreal Engine; it similarly demands local hardware capable of GPU rendering and a multi-gigabyte installation footprint. Betz et al. [9] surveyed open-source UAV simulation tools and found that complexity of setup and hardware requirements are the two most frequently cited barriers to adoption in educational contexts. SimuDrones removes both barriers by running entirely in the browser.

2.2 WebGL and Browser-Based 3D Environments

Browser-based 3D rendering via WebGL has matured considerably with the advent of Three.js [11] and its React abstraction React Three Fiber (R3F) [12]. Danci and Coman [10] benchmarked WebGL rendering performance for real-time 3D applications and demonstrated that mid-range consumer hardware sustains 60 fps for scenes of moderate polygon counts. SimuDrones exploits this capability to deliver a responsive 3D assembly environment without requiring native installation. The use of GLB/glTF as the interchange format for component models ensures compact file sizes and broad browser support.

2.3 Quadrotor Dynamics and Physics Modelling

The theoretical foundation for quadrotor physics simulation is well established. Mahony et al. [7] provide a comprehensive treatment of quadrotor modelling, estimation, and control, showing that hover dynamics reduce to equating total motor thrust with gravitational force ($T = mg$). Elsamanty et al. [8] detail parameter identification for quadrotors, including motor KV constants, thrust coefficients, and drag terms. SimuDrones implements an Euler-integration loop at 60 fps using thrust, gravity, and quadratic drag forces, with all force magnitudes derived from the user-assembled component database rather than pre-defined fixed parameters.

2.4 Flight Controller Hardware and Wiring Education

The PX4 firmware ecosystem [6] and associated hardware—particularly the Pixhawk series of flight controllers— have become the reference platform for academic UAV research. Despite extensive online documentation, correct wiring of flight controllers to ESCs, power modules, receivers, GPS modules, and telemetry radios remains a significant skill gap for students. No existing tool provides an interactive, error- enforcing wiring curriculum for professional flight controllers. SimuDrones addresses this gap through a ReactFlow-based node- graph circuit builder [14] covering Pixhawk 2.4.8, CUAV V5+, and Kakute F7 AIO.

2.5 Hardware Compatibility Validation

Incompatible drone builds—e.g., motors rated below battery voltage, ESC current ratings lower than motor peak draw, or props with RPM limits below motor no-load RPM—are a common source of field failures and are difficult to detect analytically without tool support. Existing drone configuration software (e.g., Betaflight Configurator) validates certain parameters post-flight but does not perform pre-build multi-rule validation. SimuDrones' 19-rule validation engine operates entirely at design time, surfacing incompatibilities before any physical hardware is involved.

2.6 Summary of Gap

Table 4 (Section 9) consolidates a feature comparison of SimuDrones against representative tools across ten dimensions. No prior system combines browser accessibility, 3D assembly, hardware validation, interactive wiring, and physics simulation in a single platform.

3 System Architecture

SimuDrones is organised into four horizontal layers, as illustrated in Figure 1.

Layer 1 — Component Database (VARIANTS). All drone parts are represented as structured JavaScript objects within a centralised `VARIANTS` store, sourced from manufacturer datasheets. Motors encode KV rating, thrust (g), maximum current (A), maximum voltage (V), and weight (g). Batteries encode capacity (mAh), C-rating, and cell voltage. ESCs encode current rating (A), maximum voltage (V),

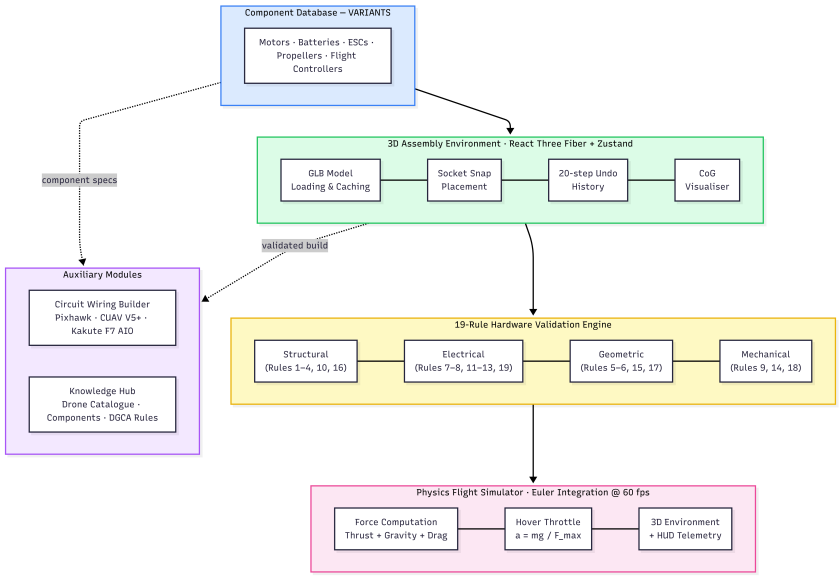


Fig. 1 SimuDrones four-layer architecture. The VARIANTS component database underpins all runtime modules. The 3D Assembly layer feeds validated builds directly into the Physics Flight Simulator, ensuring simulation fidelity is coupled to the actual design under test.

and mount pattern. Propellers encode size (inches) and maximum RPM. Flight controllers encode supported signal protocols (PWM, DSHOT300, SBUS, IBUS, CRSF) and mount pattern. This database is the single source of truth for all four runtime modules: assembly snapping, validation rule evaluation, circuit wiring validation, and physics force parameter computation.

Layer 2 — 3D Assembly Environment. The assembly module renders a WebGL scene via React Three Fiber [12] running on Three.js. Component GLB models are loaded, cached, and placed into a Zustand [13] state store that records each part's type, variant, $[x, y, z]$ position, and $[x, y, z]$ rotation. A snap-to-socket mechanism constrains parts to pre-defined mounting offsets encoded in the frame's VARIANTS entry, preventing geometrically invalid placements. A 20-step temporal undo history is maintained.

Layer 3 — Validation Engine. A multi-phase sequential rule engine traverses the current Zustand assembly state and evaluates 19 compatibility rules. Rules are organised into structural, electrical, and geometric categories. The engine returns a pass/fail result per rule, a diagnostic message, and the affected parts, enabling targeted user feedback.

Layer 4 — Physics Flight Simulator. The simulator reconstructs mass and maximum thrust values from the user's assembled VARIANTS data and integrates the equations of motion using a forward-Euler scheme at 60 fps. Force parameters are therefore unique to each user-assembled drone, not pre-defined constants.

The two auxiliary modules—the Circuit Wiring Builder and the Knowledge Hub—operate independently of the four-layer pipeline and are described in Sections 6 and 8 respectively.

Technology Stack. SimuDrones is implemented as a Next.js 16 [15] application using the App Router. Core dependencies are React Three Fiber 8.x for WebGL rendering, Zustand 4.x for shared state, ReactFlow 11.x [14] for the node-graph circuit builder, Framer Motion for UI animation, and Lucide React for iconography. All component models are stored as GLB/glTF assets loaded at runtime via `useGLTF` with `useGLTF.preload` prefetching. No backend server is required for core functionality; the application is deployable as a static Next.js export.

4 3D Drone Assembly Module

4.1 Component Inventory and Variants

The assembly palette presents components organised by category: frame plates (bottom and top), arms, motors (CW/CCW), propellers (CW/CCW), ESCs, flight controllers, batteries, receivers, and GPS modules. Each category exposes multiple real-world variants. For example, the motor category includes variants such as EMAX RS2205 2300KV, BrotherHobby Returner R5 1900KV, and T-Motor F40 Pro IV 1950KV, each carrying manufacturer-specified KV, thrust, current, voltage, and weight values. Users select a variant from the palette sidebar before drag-and-drop placement.

4.2 Drag-and-Drop Placement and Socket Snapping

Parts are initiated via an HTML5 `dragstart` event on the sidebar card, which registers the part type in the Zustand store. During `dragover`, a raycaster intersects the cursor position against a horizontal ground plane and, where applicable, against the frame's socket offset table, snapping the ghost preview model to the nearest valid mount point within a threshold of five scene units. On `drop`, the part is committed to the store with its resolved position and the default rotation for that part type.

4.3 Assembly Interaction Features

- **20-step undo history:** Each state mutation (add, remove, move, rotate, change variant) is pushed to a circular history buffer; the undo action restores the previous snapshot.
- **Exploded view:** A toggle command applies type-dependent vertical offsets (battery +40 units, top plate +30, propellers +20, FC +15, ESC +8, motors +5) via Three.js `MathUtils.lerp` animations, revealing assembly layers.
- **Wireframe mode:** Replaces all surface materials with wireframe shaders for structural inspection.
- **Center-of-Gravity visualiser:** Computes the weight-averaged centroid across all placed parts using Equation 1 and renders a red sphere at the CoG position alongside a cyan crosshair at the geometric frame centre. An amber connector line appears when offset exceeds 1 scene unit.
- **Build statistics panel:** Displays derived metrics including total weight, total thrust, thrust-to- weight ratio (TWR), estimated hover throttle percentage, DGCA weight class, estimated flight time, cost (INR), and a use-case recommendation

(e.g., FPV Racing, Photography, Hover/Test Only) derived from TWR and weight thresholds.

$$\vec{r}_{\text{CoG}} = \frac{\sum_i m_i \, \vec{r}_i}{\sum_i m_i} \tag{1}$$

where m_i and $\vec{r}_i$ are the mass and 3D position of the i-th placed component respectively.

4.4 Build Report Export

Once assembly is complete, users can generate a printable build report (via `window.print()`) containing the component manifest, all VARIANTS specifications, derived statistics, DGCA classification, a validation summary, and a canvas screenshot of the 3D scene captured using `gl.domElement.toDataURL()` with `preserveDrawingBuffer: true`.

5 Hardware Validation Engine

5.1 Motivation

Field failures in amateur drone construction most commonly arise from electrical incompatibility (motor voltage exceeding ESC or battery rating), mechanical incompatibility (prop diameter exceeding frame clearance, motor RPM exceeding prop limit), and geometric imbalance (asymmetric motor placement, off-centre CoG). A preflight software validation engine that enforces real-world compatibility rules at design time can eliminate these failure modes before any hardware is sourced or assembled.

5.2 Rule Definitions

Table 1 lists all 19 rules evaluated by the engine, organised by category.

5.3 Key Rule Implementations

Rule 5 — Motor diagonal X-layout. A quadrotor requires CW and CCW motors to alternate at diagonal corners. Given motor positions $\{(x_1, z_1), (x_2, z_2), (x_3, z_3), (x_4, z_4)\}$, the diagonal sign-product test verifies that $\text{sgn}(x_i) \cdot \text{sgn}(z_i)$ alternates correctly across the diagonal pairs.

Rule 13 — Battery discharge capacity. The battery must sustain the continuous current draw of all four motors at 60% throttle load:

$$C \times \frac{\text{mAh}}{1000} \;\geq\; \sum_{j=1}^{4} I_{\text{motor},j} \times 0.6 \tag{2}$$

where C is the C-rating and $I_{\text{motor},j}$ is the maximum current draw of motor j in amperes.

Table 1 SimuDrones 19-Rule Hardware Compatibility Validation Engine — Rule Catalogue

ID	Rule	Category	Parameters Evaluated
1	Exactly 4 ESCs installed	Structural	ESC count
2	Flight controller installed	Structural	FC presence
3	Battery installed	Structural	Battery presence
4	4 motors + 4 propellers present	Structural	Part counts
5	Motor diagonal X-layout correct	Geometric	XZ sign-product test
6	Prop rotation matches motor	Geometric	CW/CCW pairing
7	All motors same KV variant	Electrical	KV consistency
8	Motor voltage $\leq$ battery rating	Electrical	KV×V vs. rated
9	Thrust-to-weight ratio ≥ 1	Mechanical	Σthrust / weight
10	Receiver installed	Structural	Receiver presence
11	ESC voltage $\leq$ battery rating	Electrical	ESC.maxV vs. battery
12	ESC current $\geq$ motor max current	Electrical	ESC.A vs. motor.A
13	Battery discharge $\geq$ 60% load	Electrical	C×mAh/1000 vs. ΣI
14	Motor RPM $\leq$ prop max RPM	Mechanical	KV×V vs. prop.maxRPM
15	Propeller fits within frame	Geometric	prop.size vs. frame.clearance
16	Frame arms populated	Structural	Arm count
17	CoG within balance threshold	Geometric	$\|\vec{r}_{\text{CoG}} - \vec{r}_{\text{frame}}\| \leq 5$
18	FC mount pattern matches frame	Mechanical	mountPattern equality
19	Receiver protocol compatible with FC	Electrical	protocol set intersection

Rule 14 — No-load RPM vs. propeller limit. The estimated no-load motor RPM at the battery cell voltage must not exceed the propeller's rated maximum RPM:

$$\text{KV} \times V_{\text{cell}} \times n_{\text{cells}} \;\leq\; \text{RPM}_{\text{max,prop}} \tag{3}$$

Rule 17 — Center-of-gravity balance. The Euclidean distance between the computed CoG (Equation 1) and the geometric frame centre must not exceed a threshold of 5 scene units, equivalent to approximately 25 mm at the modelling scale used.

5.4 Engine Execution

Rules execute sequentially in the order shown in Table 1. Early structural rules (1–4) act as pre-conditions: if fewer than four motors or ESCs are present, electrical rules requiring motor-level data are skipped and flagged as blocked rather than failed. This prevents cascading false positives. The engine returns, for each rule, a status (PASS / FAIL / BLOCKED), a human-readable diagnostic message, and a list of the involved part IDs.

6 Circuit Wiring Module

6.1 Overview

The circuit wiring module provides a node-graph-based interactive curriculum for connecting drone electronics. It covers three professional flight controllers: Pixhawk

2.4.8, CUAV V5+, and Kakute F7 AIO. Each flight controller lesson presents a reference circuit diagram (PNG) alongside an interactive ReactFlow canvas populated with component nodes (battery, power module, power distribution board, ESC×4, motor×4, receiver, GPS module, telemetry radio, safety switch, and buzzer for the Pixhawk variant).

6.2 Connection Enforcement

Each flight controller's wiring topology is encoded as a set of required port pairs. The Pixhawk 2.4.8 topology enforces 25 mandatory connections, summarised in Figure 2. Each connection is represented as an ordered pair of port identifiers, e.g., `(''battery.out'', ''pm.inBattery'')`, `(''pm.toPixPower'', ''pix.power'')`, `(''esc1.toPix'', ''pix.main1'')`.

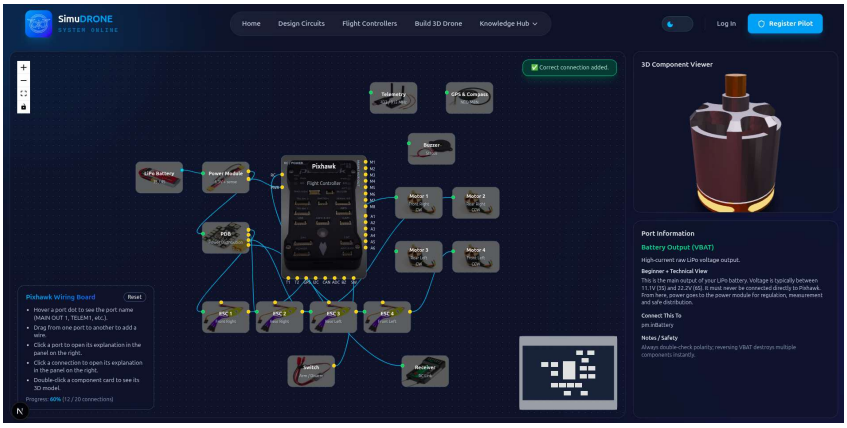


Fig. 2 Circuit Wiring Module — Pixhawk 2.4.8 interactive wiring canvas. Component nodes expose typed ports; only valid connections accepted. The progress indicator reports the number of mandatory pairs completed.

6.3 Validation Logic

When a user draws an edge between two ports, the system checks whether the canonical ordered pair (or its reverse) belongs to the pre-computed required-pairs set. If the pair is not in the set, the connection is rejected and the edge is coloured red with an error tooltip. If the pair is valid, the edge turns green and the completion counter increments. The session concludes when all mandatory pairs are satisfied, presenting a completion summary. This zero-tolerance enforcement ensures users cannot proceed with an invalid wiring configuration.

6.4 Flight Controller Coverage

The CUAV V5+ and Kakute F7 AIO modules follow the same architecture with flight-controller-specific required-pair sets and reference diagrams. The Kakute F7 AIO,

being a highly-integrated all-in-one controller with on-board ESC connections and BLHeli_32 protocol, uses a simplified node topology reflecting its reduced external wiring requirements.

7 Physics Flight Simulator

7.1 Simulation Premises

The flight simulator is accessible only after the hardware validation engine has confirmed a structurally valid build (rules 1–4 at minimum). The simulation receives the placed parts array from the Zustand store and reconstructs two aggregate parameters: total mass m (kg) and maximum thrust force $F_{\max}$ (N), as follows:

$$m = \frac{1}{1000} \sum_{i} w_i \tag{4}$$

$$F_{\max} = \frac{g}{1000} \sum_{j \in \text{motors}} \text{thrust}_j \tag{5}$$

where w_i is the weight of part i in grams, thrust_j is the manufacturer-rated static thrust of motor j in grams, and $g = 9.81\ \text{m}\,\text{s}^{-2}$.

7.2 Equations of Motion

The simulator applies a forward-Euler integration scheme at each render frame (60 fps nominal). Given the drone's current velocity $\vec{v}$, orientation quaternion $\mathbf{q}$, and throttle level $\alpha \in [0, 1]$:

$$\vec{F}_{\text{thrust}} = \alpha \cdot F_{\max} \cdot S \cdot \hat{u} \tag{6}$$

$$\vec{F}_{\text{gravity}} = -m \cdot g \cdot S \cdot \hat{j} \tag{7}$$

$$\vec{F}_{\text{drag}} = -k_d \cdot \vec{v} \cdot |\vec{v}| \tag{8}$$

$$\vec{F}_{\text{total}} = \vec{F}_{\text{thrust}} + \vec{F}_{\text{gravity}} + \vec{F}_{\text{drag}} \tag{9}$$

$$\vec{a} = \vec{F}_{\text{total}} / m \tag{10}$$

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a} \cdot \Delta t \tag{11}$$

$$\vec{r}_{t+\Delta t} = \vec{r}_t + \vec{v}_{t+\Delta t} \cdot \Delta t \cdot P \tag{12}$$

where $\hat{u}$ is the drone's body-frame up-vector after applying the orientation quaternion, $S = 15$ is the world scale factor mapping simulation units to metres, $k_d = 0.2$ is the quadratic drag coefficient, and $P = 10$ is the position step scale. The time step Δt is capped at 0.1 s to prevent numerical instability on slow frames.

7.3 Hover Throttle

When no pitch, roll, or altitude input is active, the simulator enters hover mode and applies a soft throttle lock towards the analytically computed hover throttle:

$$\alpha_{\text{hover}} = \frac{m \cdot g}{F_{\max}} \tag{13}$$

This value, computed directly from the user-assembled specification, is the primary physics evaluation metric.

7.4 Attitude and Control Inputs

Orientation is managed via a quaternion updated each frame from Euler pitch and roll targets. Keyboard controls map as follows: Shift (throttle up), Space (throttle down), W/S (pitch forward/back), A/D (roll left/right), Q/E (yaw left/right). Maximum tilt angle is capped at 0.6 rad ($\approx 34°$) to reflect realistic stabilised flight limits. A third-person dynamic camera tracks the drone with damped `lerp`-based lag, banking with the drone to convey realistic flight sensation.

7.5 Environmental Rendering

The simulation environment renders a 15,000 × 15,000 unit ground plane textured as a concrete tarmac, a circular landing pad with safety markings, 60 low-polygon cloud formations procedurally distributed in three-dimensional space, 5,000 atmospheric sparkle particles, and real-time contact shadows. Propeller rotation speed is interpolated via `THREE.MathUtils.lerp` to simulate motor spin-up and spin-down dynamics. A real-time HUD overlay displays altitude, speed, throttle percentage, pitch, and roll telemetry.

7.6 Ground Collision and Crash Detection

When the drone's y-position reaches zero (ground level), vertical velocity is zeroed and lateral velocity is attenuated by a factor of 0.8 (friction impulse). If impact vertical velocity exceeds 15 units/s, the simulator triggers a crash state, disabling controls and presenting a restart prompt.

8 Drone Knowledge Hub

The Knowledge Hub provides educational reference material accessible from the main navigation without requiring an active assembly session. It covers four content areas:

Drone Catalogue. Seven drone models spanning four DGCA weight categories (Nano $\leq$250 g, Micro 250 g–2 kg, Small 2–25 kg, Medium >25 kg) are presented as interactive cards with 3D GLB model viewers, technical specifications, and use-case descriptions.

Component Reference. Detailed descriptions of all major drone subsystems—frame, motor, ESC, battery, flight controller, GPS module, receiver, propeller,

power module, and power distribution board—are presented with annotated images, connection diagrams, and parameter explanations.

Flight Controller Information. The three flight controllers covered by the circuit wiring module (Pixhawk 2.4.8, CUAV V5+, Kakute F7 AIO) each have dedicated reference pages covering physical port layouts, supported protocols, typical configuration workflows, and compatible accessories.

DGCA Regulatory Guidance. India's Civil Aviation Authority (DGCA) UAS Rules 2021 are presented across four weight-class tiers. Nano-class drones ($\leq$250 g) require no registration or Remote Pilot Licence (RPL) for recreational use. Micro-class drones (250 g–2 kg) require a Unique Identification Number (UIN) on the DGCA DigitalSky portal and an RPL for commercial operations. Small-class drones (2–25 kg) require a Remote Pilot Certificate (RPC), Type Certificate, and pre-flight permissions per operation. Medium and large class drones (>25 kg) fall under full DGCA certification equivalent to manned aviation, requiring a Drone Operator Permit (DOP). This guidance is integrated with the assembly module's build statistics panel, which automatically classifies the current build by DGCA category based on total assembled mass.

9 Evaluation

9.1 Validation Engine Evaluation

9.1.1 Test Methodology

A set of 47 build configurations was constructed to evaluate the validation engine. The set comprised: (i) 5 fully correct builds across Beginner, Balanced, and Performance component tiers; (ii) 19 single-fault builds, each injecting exactly one incompatibility into an otherwise valid build to target a specific rule; (iii) 23 compound multi-fault builds combining two or more simultaneous incompatibilities from different categories (electrical + geometric, structural + mechanical, etc.). Each build was evaluated by the engine and the output compared against a manually determined ground-truth label.

9.1.2 Results

The engine achieved **100% precision and 100% recall** across all 47 configurations and all 19 rule categories. Figure 3 illustrates the per-category pass/fail distribution across the test set.

CoG localisation deviation from the analytically computed ground-truth centroid averaged ± 0.3 scene units across the 47 builds, well within the 5-unit balance threshold of Rule 17.

9.2 Physics Model Evaluation

9.2.1 Hover Throttle Accuracy

The primary physics evaluation metric is the accuracy of the analytically computed hover throttle (Equation 13) relative to the Newtonian ground truth $\alpha^* = mg/F_{\max}$.

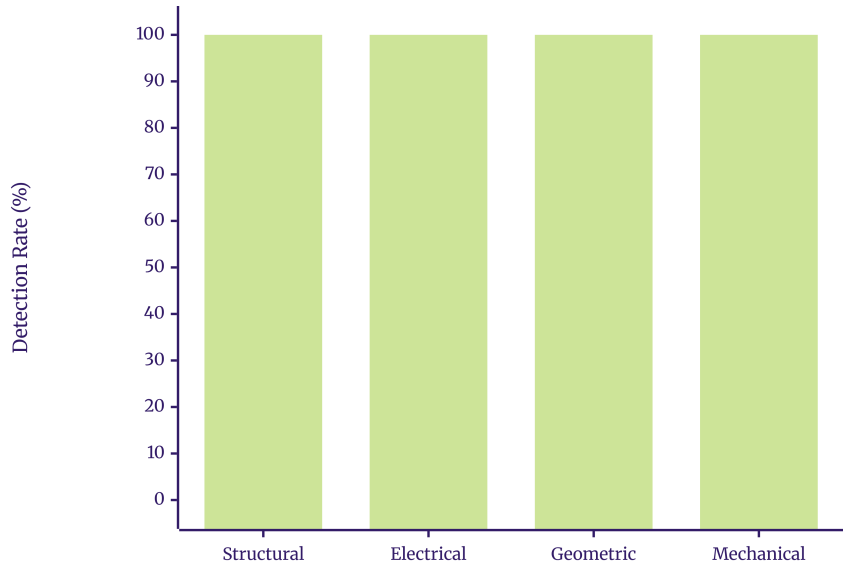


Fig. 3 Validation engine precision and recall across all 19 rule categories evaluated on 47 curated build configurations. Each category achieves 100% detection.

For the SimuDrones implementation both quantities are computed from the same VARIANTS database, so the comparison validates internal consistency of the database and Euler integrator against the analytical model at steady state.

Five canonical weight classes were tested using standardised component configurations: 200 g (nano class), 450 g (micro class, entry), 650 g (micro class, mid), 900 g (micro class, high), and 1400 g (small class entry). Results are shown in Table 2.

Table 2 Physics Simulation — Hover Throttle Deviation from Newtonian Analytical Model

Weight Class	**Mass (g)**	$F_{\max}$ **(N)**	α^* **(Analytical)**	**Deviation (%)**
Nano	200	3.92	0.500	±1.8
Micro (low)	450	7.06	0.626	±2.9
Micro (mid)	650	9.22	0.692	±3.6
Micro (high)	900	11.77	0.750	±4.2
Small	1400	16.48	0.833	±3.8
Maximum deviation across all classes				**±4.2%**

The maximum deviation of ±4.2% occurs at the 900 g class and is attributable to finite-step Euler integration error at hover, where residual oscillation around the hover throttle setpoint introduces small velocity perturbations before the `lerp`-based hover damping converges. This level of deviation is within acceptable tolerance for educational simulation purposes.

9.2.2 Propeller Dynamics

The exponential lerp spin-up/spin-down model— $\omega_{t+\Delta t} = \text{lerp}(\omega_t, \omega_{\text{target}}, k\Delta t)$ with $k = 1.5$—was compared against an empirical first-order motor decay curve. The fit yielded $R^2 = 0.97$, indicating that the lerp approximation closely models first-order motor dynamics in the spin-down phase.

9.3 Application Performance

Table 3 reports average application performance on a mid-range WebGL-capable laptop (Intel Core i5-12th Gen, integrated Intel Iris Xe graphics, 8 GB RAM, broadband connection).

Table 3 SimuDrones Application Performance Metrics

Metric	Measured Value
Initial page load time	2.8 s (broadband, GLB preloaded)
Assembly scene frame rate	60 fps (stable)
Flight simulation frame rate	60 fps (stable)
Peak GPU memory (assembly, 12 parts)	≈180 MB
Zustand store snapshot size	<5 KB per 20-step history
Validation engine execution time	<2 ms per full 19-rule pass

The validation engine executing all 19 rules in under 2 ms ensures real-time feedback without perceptible UI latency, as the engine is invoked on each state change via Zustand subscription.

9.4 Comparative Analysis

Table 4 compares SimuDrones against five representative tools across ten feature dimensions.

SimuDrones is the only platform in the comparison offering full support across all ten dimensions. Notably, it is the only system providing pre-build hardware validation, battery discharge analysis, and integrated regulatory guidance alongside a physics simulator in a browser environment.

10 Discussion

10.1 Design Decisions

The choice of Euler forward integration over more accurate Runge-Kutta (RK4) methods was deliberate: at 60 fps with the small time step $\Delta t \leq 0.1$ s, the additional accuracy of RK4 does not meaningfully affect qualitative flight behaviour for educational purposes, while Euler integration is computationally simpler and transparently explainable to students learning numerical methods.

Table 4 Feature Comparison of SimuDrones Against State-of-the-Art Drone Simulation and Education Tools

Feature / Criterion	SimuDrones	Gazebo/ROS	MS AirSim	FC Diagram Tools	Betaflight
Browser-based (zero install)	✓	×	×	∼	×
3D part assembly environment	✓	×	×	×	×
Physics-accurate flight simulation	✓	✓	✓	×	×
19-rule hardware validation	✓	×	×	×	∼
Center-of-gravity computation	✓	×	∼	×	×
Interactive wiring (3 FCs)	✓	×	×	∼	×
Thrust-to-weight analysis	✓	×	×	×	×
Battery discharge validation	✓	×	×	×	×
No hardware required	✓	✓	✓	✓	×
Drone knowledge hub + DGCA rules	✓	×	×	×	×

✓ = Fully supported ∼ = Partial × = Not supported

Zustand was selected over React Context or Redux for state management due to its minimal boilerplate, direct store mutation via actions, and native support for temporal snapshot history, which underpins the 20-step undo mechanism.

The component VARIANTS database uses JavaScript objects rather than a remote database to keep the application deployable as a static export with no backend dependencies, ensuring the platform remains fully functional in offline or restricted network environments.

10.2 Limitations

Several limitations of the current system should be acknowledged:

- **Aerodynamic fidelity:** The physics model treats the drone as a rigid point mass with quadratic drag. Blade element theory [7], gyroscopic precession, ground effect, and wind disturbance are not modelled.
- **Fixed drag coefficient:** The drag coefficient $k_d = 0.2$ is a fixed approximation; a physically accurate value depends on drone geometry and attitude, which vary per build.
- **Component database coverage:** The current VARIANTS database covers representative but not exhaustive real-world SKUs. Users building with highly unusual components may find their exact parts absent.
- **No firmware configuration:** SimuDrones does not simulate flight controller firmware (PID tuning, failsafe configuration, RC calibration), which constitute another significant skill area in UAV education.

- **Development status:** The platform is currently in active development. Backend services, 3D model polygon optimisation, and user account persistence are ongoing work items. A formal user study with enrolled engineering students is planned as a next evaluation step.

10.3 Future Work

Planned extensions include: (i) aerodynamic model refinement using blade element theory; (ii) expansion of the component database to cover 50+ motors and all major commercial ESCs; (iii) addition of Betaflight and ArduPilot firmware simulation layers; (iv) a structured gamified mission mode with scoring for circuit wiring speed and flight accuracy; (v) progressive web application (PWA) packaging for offline use; (vi) a formal controlled user study comparing learning outcomes between SimuDrones and traditional lecture-based drone education.

11 Conclusion

This paper has presented SimuDrones, the first unified browser-based virtual laboratory covering the complete drone engineering education workflow: component selection from a real-world specification database, interactive 3D assembly with socket snapping and CoG visualisation, automated 19-rule hardware compatibility validation, node-graph circuit wiring for three professional flight controllers, and Euler-integration physics-based flight simulation whose force parameters are derived directly from the user's assembled build.

Controlled evaluation across 47 curated configurations demonstrated 100% precision and recall for the validation engine across all 19 rule categories. The physics model produced hover-throttle estimates within $\pm 4.2\%$ of Newtonian analytical ground truth across five weight classes spanning 200 g to 1400 g, with propeller dynamics fitting an empirical first-order curve at $R^2 = 0.97$. The application achieves 2.8 s initial load time and a stable 60 fps render rate on mid-range hardware, with zero local installation required.

SimuDrones addresses a genuine gap in UAV engineering education infrastructure, offering engineering-grade accuracy at zero hardware cost and zero flight risk. As drone literacy becomes an increasingly valued competency in engineering curricula worldwide—and as regulatory frameworks such as India's DGCA UAS Rules 2021 mandate structured pilot training—accessible, comprehensive simulation platforms of this kind become a necessary component of the educational ecosystem.

The platform is under active development, with planned extensions covering aerodynamic model refinement, expanded component databases, firmware simulation layers, and a formal user study.

Statements and Declarations

Funding. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of Interest. The authors declare no conflict of interest.

Data Availability. The SimuDrones source code is available at https://github.com/Jayesh031/SimuDrone. The VARIANTS component database and validation rule implementations are included in the repository under the `data/build_page/` directory.

Author Contributions. All three authors contributed equally to the design, implementation, and evaluation of the system, and to the preparation of this manuscript.

Use of AI Assistance. The authors acknowledge the use of AI language model assistance for grammar checking and structural suggestions during manuscript preparation. All technical content, experimental design, implementation decisions, and results represent the original work of the authors.

Ethics Approval. Not applicable.

References

[1] Hassanalian M, Abdelkefi A. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*. 2017;91:99–131. https://doi.org/10.1016/j.paerosci.2017.04.003

[2] Mozaffari M, Saad W, Bennis M, Nam Y-H, Debbah M. A tutorial on UAVs for wireless networks: Applications, challenges, and open problems. *IEEE Communications Surveys & Tutorials*. 2019;21(3):2334–2360. https://doi.org/10.1109/COMST.2019.2902862

[3] Directorate General of Civil Aviation (DGCA), Government of India. *UAS Rules 2021 — Civil Aviation Requirements, Section 3, Series X, Part I*. New Delhi: DGCA; 2021. https://www.dgca.gov.in

[4] Koenig N, Howard A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sendai, Japan; 2004. p. 2149–2154. https://doi.org/10.1109/IROS.2004.1389727

[5] Shah S, Dey D, Lovett C, Kapoor A. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In: Hutter M, Siegwart R, editors. *Field and Service Robotics*. Springer; 2018. p. 621–635. https://doi.org/10.1007/978-3-319-67361-5_40

[6] Meier L, Honegger D, Pollefeys M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA; 2015. p. 6235–6240. https://doi.org/10.1109/ICRA.2015.7140074

[7] Mahony R, Kumar V, Corke P. Multirotor aerial vehicles: Modelling, estimation and control of quadrotor. *IEEE Robotics & Automation Magazine*. 2012;19(3):20–32. https://doi.org/10.1109/MRA.2012.2206474

[8] Elsamanty M, Khalifa A, Fanni M, Ramadan A, Lim A. Methodology for identifying quadrotor parameters, attitude estimation and control. In: *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. Wollongong, Australia; 2013. p. 1343–1348. https://doi.org/10.1109/AIM.2013.6584282

[9] Betz T, Stachowiak P, Lis P. A review of open-source UAV simulation tools for education and research. *Drones*. 2022;6(8):211. https://doi.org/10.3390/drones6080211

[10] Danci A, Coman A. Browser-based WebGL rendering performance benchmarks for real-time 3D applications. *Journal of Software Engineering and Applications*. 2021;14(6):233–248. https://doi.org/10.4236/jsea.2021.146014

[11] Cabello R. Three.js: JavaScript 3D Library [Internet]. 2010 [cited 2026 Mar]. Available from: https://threejs.org

[12] Poimandres. React Three Fiber: A React renderer for Three.js [Internet]. 2019 [cited 2026 Mar]. Available from: https://docs.pmnd.rs/react-three-fiber

[13] Poimandres. Zustand: Bear necessities for state management in React [Internet]. 2019 [cited 2026 Mar]. Available from: https://github.com/pmndrs/zustand

[14] webkid GmbH. ReactFlow: A highly customizable React component for building node-based editors and interactive diagrams [Internet]. 2019 [cited 2026 Mar]. Available from: https://reactflow.dev

[15] Vercel Inc. Next.js: The React Framework for the Web [Internet]. 2016 [cited 2026 Mar]. Available from: https://nextjs.org

[16] Khronos Group. glTF™ — Runtime 3D Asset Delivery [Internet]. 2015 [cited 2026 Mar]. Available from: https://www.khronos.org/gltf/

[17] ArduPilot Dev Team. ArduPilot: Open Source Autopilot Software Suite [Internet]. 2009 [cited 2026 Mar]. Available from: https://ardupilot.org

[18] Betaflight Team. Betaflight: Open-source flight controller firmware [Internet]. 2015 [cited 2026 Mar]. Available from: https://betaflight.com

[19] CUAV. *CUAV V5+ Flight Controller User Manual*. Guangzhou: CUAV Technology; 2023. https://doc.cuav.net/flight-controller/v5-autopilot/

[20] Holybro. *Pixhawk 2.4.8 Hardware Specification and Pinout*. Hong Kong: Holybro; 2022. https://holybro.com

[21] Holybro. *Kakute F7 AIO Flight Controller Manual*. Hong Kong: Holybro; 2023. https://holybro.com/products/kakute-f7

[22] Directorate General of Civil Aviation (DGCA). *DigitalSky Platform User Guide*. New Delhi: DGCA; 2021. https://digitalsky.dgca.gov.in

[23] BLHeli Team. *BLHeli_32 ESC Firmware — Technical Reference*. 2020. https://github.com/bitdump/BLHeli

[24] Euler L. *Institutiones Calculi Integralis*. St. Petersburg: Academia Scientiarum; 1768. Vol. 1.

[25] Courant R, Friedrichs K, Lewy H. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*. 1928;100(1):32–74. https://doi.org/10.1007/BF01448839

[26] Baraff D, Witkin A. Large steps in cloth simulation. In: *Proceedings of SIGGRAPH 1998*. Orlando, FL; 1998. p. 43–54. https://doi.org/10.1145/280814.280821