# Systems: A Sandbox for Autonomous Navigation Algorithms

Skytag Open | University student competition | 10 October 2026

**Research affiliation:** Department of Computer Information Technologies (KIT), Kyiv Aviation Institute
**Student competition for:** École Nationale de l'Aviation Civile (ENAC) students

# 1. A sandbox for autonomous navigation

## Abstract

Systems is a university student algorithm simulation competition for students of École Nationale de l'Aviation Civile (ENAC), with research affiliation to the Department of Computer Information Technologies (KIT), Kyiv Aviation Institute. Student teams create autonomous navigation software and test it against other submissions in a shared virtual game. Every eligible scored agent must control a flight-controller software-in-the-loop (SITL) instance through MAVLink. The main game is Tag, Follow & Catch: one agent flies away while the other follows and attempts a virtual catch. Checkpoint Race is a separate game for ordered route completion and obstacle avoidance. A trusted referee evaluates simulator state while participants receive only the declared observations. This paper specifies the architecture, scoring, evaluation protocol and reporting requirements. The paper includes the organizer-confirmed 2025 standings and analytical plots of scoring and observation timing.

## Scope and contribution

Skytag Open is the main chase-and-escape game within Systems. Checkpoint Race adds a second game on the same runtime. This paper documents their competition contract and reporting methodology. Users submit software, not physical vehicles. Guidance refers here to route following; targeting refers only to selecting virtual game objectives. The principal research questions concern autonomous navigation: completing routes, responding to moving agents, maintaining clearance and recovering from stale state estimates.

## Why require a flight controller?

A direct position-update game can reward policies that ignore estimation delay, tracking limits and control-mode transitions. Requiring the SITL control loop makes those interfaces part of the benchmark. PX4 and ArduPilot document software simulation of their flight stacks [1,2]. Systems uses separate, version-pinned tracks for these stacks, each requiring adapter qualification. SITL improves interface fidelity but is not evidence of physical-world performance.

## Reporting scope

The 2025 standings are the original eight-team results confirmed by the competition organizer. They are preserved as reported season totals. The current normalized per-heat scoring rules are documented separately and are not used to recalculate those historical totals. Per-heat telemetry is not included in this archive, so the paper does not infer unreported navigation errors or statistical significance from the standings. The archive contains the paper, standings data, plot builder and website.

# 2. Runtime and MAVLink contract

## Required control path

Each submission executes in an isolated process or container with a private connection to its own trusted MAVLink gateway and SITL flight controller. The simulator generates sensor inputs and applies controller actuator outputs to the virtual vehicle. Algorithms must not write simulator poses or actuator state directly. Two such agents share the same world. A separate referee can read ground truth and decide game events. The simulator-to-controller transport is stack-specific; the mandatory MAVLink boundary is the participant control interface [1].

## Commands, telemetry and setpoints

The gateway must support the approved lifecycle commands through COMMAND_LONG or COMMAND_INT, record COMMAND_ACK results, and verify actual mode/state from telemetry. An accepted command is not proof of completed execution [3]. Navigation uses supported MAVLink position or velocity setpoint messages, such as SET_POSITION_TARGET_LOCAL_NED, with explicit frames, units and type masks [5]. These streamed setpoints are not individually acknowledged command transactions. HEARTBEAT, local position and attitude telemetry provide the declared vehicle state; opponent observations use a separately specified game sensor feed.

## Timing and isolation

The multirotor track uses a 20 Hz algorithm decision budget and a 90 s simulation-time heat. These are benchmark settings. PX4 Offboard has its own continuous proof-of-life requirements and loss behavior [4]; a HEARTBEAT alone is insufficient. Qualification must test the pinned firmware, stream rate, startup sequence and stale-command behavior. Restrict egress, cap CPU and memory, allowlist messages, bind each connection to one system/component identity and deny access to opponent processes and referee truth.

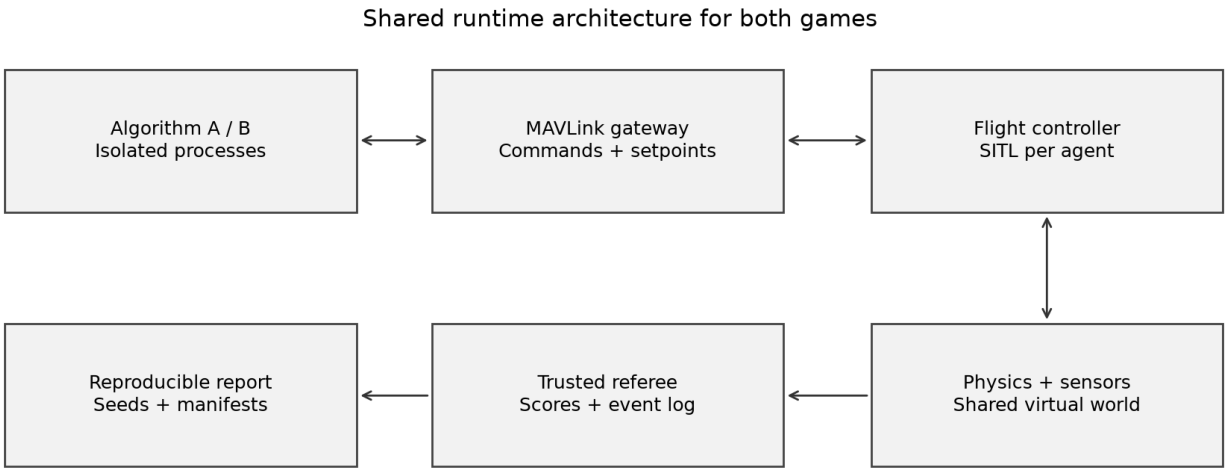



Figure 1. Shared runtime boundary. Both games use the same SITL and MAVLink contract.

# 3. Game catalog and scoring

### Main game: Tag, Follow & Catch

The runner flies away and chooses its own route. The chaser follows its observations and tries to catch the runner. There are no mandatory checkpoints in this game. A catch requires separation of at most 2 m for one continuous simulation second without contact. A catch ends the heat. The runner gets a 5 s head start before the 90 s scored clock. A timeout without a catch is an escape. Reset the world and both controllers, then swap roles and spawn sides for the return heat.

### Main-game points

For a valid catch at scored time t, the chaser receives 100 (110 - t) / 110 points; otherwise it receives 0. The runner receives 100 t_end / 90 points. A full 90 s escape is worth 100 runner points. Rank algorithms by mean score over equal numbers of runner and chaser heats. Catch rate, conditional catch time, escape rate and failure rate accompany points. These current normalized scores are distinct from archived 2025 season totals.

### Second game: Checkpoint Race

Both agents navigate five ordered checkpoints through obstacles in a shared arena. There is no tag objective. Each agent stops accumulating route time and distance when it completes the route; the heat ends when both finish or after 90 s. Swap spawn sides in the return heat. Score S = 100 V (0.80 C + 0.20 E), where C is the completed checkpoint fraction, and E = min(1, L_ref / L) for a completed route, otherwise zero. L_ref is the published collision-free static reference route. Earlier completion breaks equal-score ties; incomplete episodes use the 90 s time cap.

### Shared validity rules

Use a 100 x 60 x 30 m arena and identical vehicle capabilities within each track. An arena exit, resource-budget violation or agent contract failure scores zero for that agent and ends the heat. A collision invalidates both. A valid opponent keeps its accrued game score. Infrastructure failure or missing referee data triggers a replay with the same manifest. Failures take precedence over simultaneous catches or checkpoint events. Keep rankings separate by game and vehicle/firmware track.

### Why separate games?

Chase-and-escape tests responsive navigation against an active opponent. Checkpoint Race tests route completion and efficiency. Combining them into a checkpoint-dominated score would obscure the main game objective. Both still demand state estimation, obstacle awareness, feasible movement and reliable MAVLink control. Report metrics appropriate to each objective rather than presenting one blended leaderboard.

# 4. Confirmed season results

## 2025 Tag, Follow & Catch

The competition organizer confirmed the original eight-team standings below. Vortex Hounds leads with 1,130 season points and seven reported tags. Paper Falcon records the most runner survival time, 561 s. Results retain the original virtual vehicle classes and season point scale.

| Team | Tags / 7 | Fastest tag | Survival | Season points |
|---|---|---|---|---|
| Vortex Hounds | 7 | 8.2 s | 528 s | 1130 |
| Paper Falcon | 6 | 11.6 s | 561 s | 1035 |
| Redline Swarm | 6 | 14.9 s | 512 s | 944 |
| Tinwing Collective | 5 | 13.4 s | 498 s | 873 |
| Magpie Works | 5 | 19.7 s | 471 s | 801 |
| Bramble FPV | 4 | 16.3 s | 455 s | 731 |
| Stall Warning | 3 | 22.8 s | 402 s | 591 |
| Groundhog Dynamics | 2 | 31.5 s | 366 s | 470 |

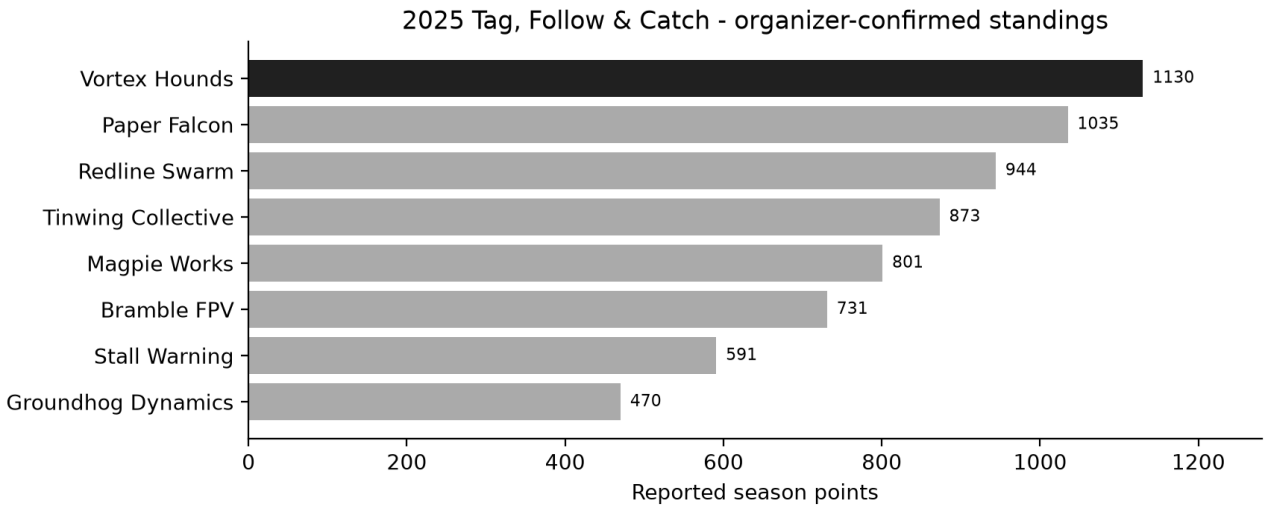



Figure 2. Organizer-confirmed 2025 season points, preserved on their archived scale.

## Reading the results

The table and plot preserve supplied aggregates. They do not reconstruct individual matches or attribute score differences to a specific navigation method. Historical season totals use their archived point scale; current normalized per-heat scores and Checkpoint Race results belong in separate leaderboards. Per-heat command traces and referee events are needed for a deeper behavior analysis.

# 5. Evaluation plan and timing analysis

## Autonomous navigation experiments

For the main game, evaluate catch rate, time to catch among successful heats, escape rate and uncaught duration. For Checkpoint Race, evaluate completion, finish time and path efficiency. Both games report collision and boundary failures, clearance, localization error, recovery time and algorithm runtime. Use held-out maps with obstacle layouts, sensor noise, observation delays and moving agents. Compare a waypoint baseline, a reactive navigator and a route planner under the same observation and compute budgets. These are comparison categories; the standings do not identify participant implementations. Keep flight-controller tuning and vehicle models fixed within each track.

## Paired protocol

A paired evaluation plan uses 30 held-out seeds per scenario, three scenarios, three algorithms and two role assignments per pair: 540 heats and 1,080 agent episodes per game. Each algorithm receives 360 episodes per game. Across both games this is 1,080 heats and 2,160 agent episodes. These are evaluation-plan counts, not the size of the archived season. Use matched world seeds, spawn swaps and opponent schedules for comparisons; keep training maps and seeds separate. Bootstrap differences by seed block, retaining paired heats and all algorithms within a block. Report interval method, sample count, failures, and per-scenario results. Seeds alone do not guarantee bitwise determinism: retain scheduler, hardware and timing configuration too.

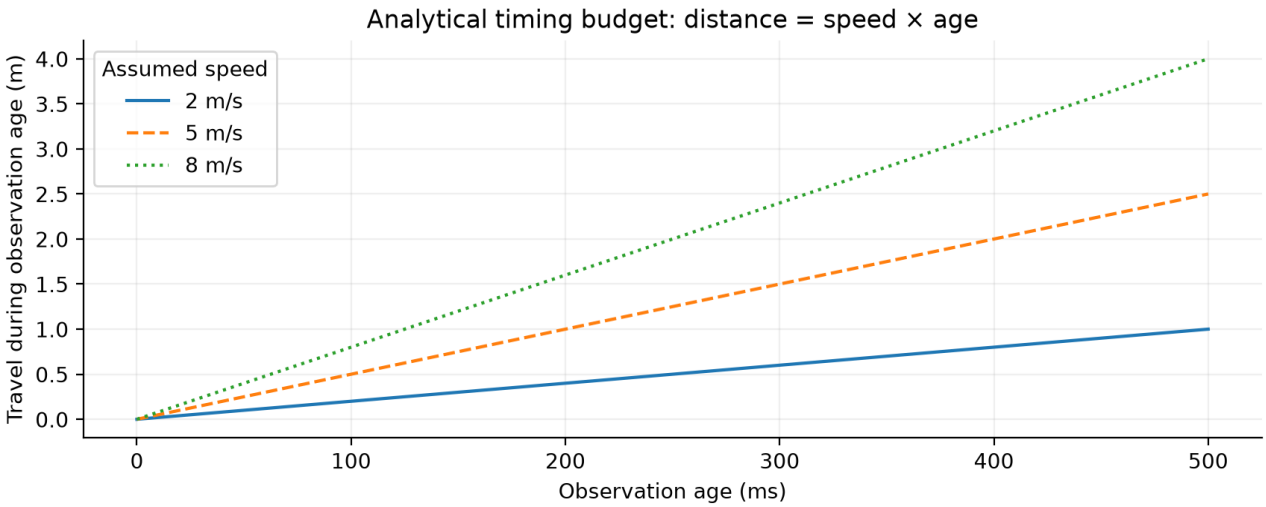



Figure 3. Analytical travel distance during observation age at three assumed speeds.

## Reasoning from the timing budget

For constant speed v and observation age d, travel during the observation gap is v times d. At an assumed 5 m/s and 200 ms, that distance is 1 m. The plot is an analytical calculation, not a measured tracking error or a collision-risk estimate. It motivates testing stale observations, prediction and replanning frequency. Localization uncertainty, acceleration, braking distance and controller lag would require additional models or measurements; the linear plot does not account for them.

# 6. Reproducibility, limitations and references

## Run manifest and acceptance gate

Record the submission image digest, source revision, firmware commit, parameter-file hash, simulator and vehicle-model versions, map hash, random seeds, game identifier, scoring-rule version, scenario settings, message allowlist, coordinate frame, clock policy and resource budget. Archive algorithm stdout, timestamped MAVLink messages and command acknowledgements, controller logs, ground-truth referee events and score components. Before accepting entries, test reset isolation, malformed messages, delayed telemetry, lost setpoint streams and resource exhaustion. Replay identical manifests and quantify any divergence.

## Interpretation and further evaluation

This repository packages the website, competition rules and report. Historical aggregates alone cannot isolate the effects of planning, perception or flight-controller tuning. Per-heat logs and matched experiments are needed for those comparisons. SITL fidelity does not establish physical-world transfer. Add vehicle types or control stacks as separately qualified tracks, and report each game independently. The archive does not include the competition execution service.

## References


[1] PX4. Simulation. https://docs.px4.io/main/en/simulation/

[2] ArduPilot. SITL Simulator (Software in the Loop). https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[3] MAVLink. Command Protocol. https://mavlink.io/en/services/command.html

[4] PX4. Offboard Mode. https://docs.px4.io/main/en/flight_modes/offboard

[5] MAVLink. Common Message Set, SET_POSITION_TARGET_LOCAL_NED. https://mavlink.io/en/messages/common.html#SET_POSITION_TARGET_LOCAL_NED


Documentation accessed 10 October 2026. These references document the controller interfaces.

## Archive contents

systems-simulation-paper.pdf; index.html; reports/paper.html; reports/summary.json; reports/season-2025.csv; four plots in PNG and SVG; scripts/build_report.py; requirements-report.txt; README.md. The builder rebuilds the standings, plots, paper and archive. The archive is downloadable from the website report section. No external publication is implied.