Software

Rebuilding the system around new rules

Two rule changes for the 2026 season invalidated the software architecture we used on our previous aircraft. All delivery payloads must now be carried simultaneously, with no landing to reload, so a target has to be located during flight instead of after recovery. The imagery map also has to be submitted within the 30-minute mission window. Our earlier system recorded video to an SD card and processed everything on the ground after landing, so we had to redesign it from the ground up.

Moving the computation onboard

We compared three architectures: processing recorded footage on the ground, streaming full frames down to a ground workstation, and running the entire pipeline onboard. Ground processing cannot designate targets in flight at all. Streaming full-resolution imagery would require a dedicated high-bandwidth link, since our telemetry radio operates two orders of magnitude below that requirement. We chose onboard processing despite its integration cost, and kept a GoPro recording continuously to local storage as a fallback. If the onboard pipeline fails, we can still process that footage after landing and preserve partial credit on the mapping task.

The eye of the aircraft

The main mission sensor is an AR0234-based global-shutter camera, mounted rigidly in nadir orientation. Global shutter was a hard requirement here. At survey speed a rolling-shutter sensor introduces a geometric skew into every frame, which corrupts both map alignment and target geolocation. We chose a fixed mount over a gimbal to eliminate a mechanical failure mode, and compensate for attitude in software instead, using the autopilot's orientation estimate. Camera intrinsics and lens distortion are recovered through checkerboard calibration, and all projection maths uses the calibrated focal length instead of the nominal lens specification.

Planning the survey

We derived the survey parameters from the sensor model instead of adopting defaults. At 60 m above ground level the camera resolves 5.0 cm per pixel across a 96 m swath. An 80% side overlap places flight lines 19.2 m apart, which covers the 150 m width of the search area in four lines, with an exposure every 12 m. Flying at twice that altitude would have cut the mapping workload fourfold, but the same imagery has to support detection of a partially occluded mannequin, so we accepted the higher processing cost to keep the ground resolution. One geometric conflict remains. The competition mandates a 150 ft minimum turn radius, far wider than our line spacing, so each pass is extended beyond the search boundary and the turn is flown in the surrounding airspace.

The onboard computer

Perception runs on an NVIDIA Jetson Orin Nano in its 25 W power mode, supplied through a dedicated regulator and actively cooled by air drawn through a fuselage inlet. We chose it over a Raspberry Pi with an NPU accelerator because of workload

heterogeneity, not raw inference speed. The mission requires neural inference and classical feature-based image registration to run concurrently, and only the CUDA architecture accelerates both. Detection and mapping share the compute node, with detection holding scheduling priority: a missed target cannot be recovered without another pass, while mapping tolerates a temporary backlog because frames are buffered.

Three separate radio links

The aircraft carries three independent radio links with separated functions: a 2.4 GHz ExpressLRS control link, a 900 MHz mLRS telemetry link, and a 5.8 GHz data link carrying detection results and image crops to the ground station. The band separation is intentional, since co-locating the data link with the control link would put the highest-bandwidth and the most safety-critical service in the same spectrum. We only transmit detection metadata and small crops, so the bandwidth requirement stays modest. Losing this link degrades the mission to autonomous release without operator confirmation, and does not cause task failure.

Finding the target

Detection uses an Ultralytics YOLO26 model, trained on our own workstation and exported to TensorRT for FP16 inference on the Jetson. Our training data combines publicly available aerial imagery with footage captured from our own aircraft. We only use nadir and near-nadir imagery, because oblique and ground-level photographs of the same objects present a fundamentally different appearance. Inference runs at 10 to 15 fps instead of at full sensor rate, which still yields around twenty independent looks at each target while leaving compute headroom for mapping. The 80% forward overlap makes every ground object appear in roughly five consecutive frames, so we do not treat raw detections as targets. Each one is georeferenced individually, clustered in ground coordinates, and promoted to a candidate based on observation count and aggregated confidence. This is what makes the system robust against the debris scattered across the search area, since the real task is selecting the single best-supported candidate for each class.

From pixel to coordinates

Every detection is converted into a ground position by undistorting the pixel, forming the corresponding camera ray, rotating it into the navigation frame using the autopilot's attitude estimate, and intersecting it with the ground surface. We built an explicit error budget for this chain, and it showed that terrain elevation dominates everything else. Field elevation across the competition area varies by roughly 35 m, and for a detection near the frame edge an unmodelled terrain error translates into a horizontal position error larger than every other term combined. Time synchronisation turned out to matter far more than we expected as well. At survey speed, 50 ms of unmodelled latency displaces a georeferenced point by over a metre, so we record frame timestamps against the autopilot's GPS-disciplined clock.

Building the map in flight

The mosaic is assembled incrementally. Each frame is registered against the accumulated map as it arrives, so a submittable product exists at every point during the flight and post-landing processing time approaches zero. The registration is not performed blind. We predict each frame's approximate ground footprint from the GNSS position and attitude recorded with it, and that prediction seeds a homography estimate refined with ORB feature matching and RANSAC outlier rejection. Telemetry seeding is essential over the largely uniform grass of the competition site, where purely appearance-based matching is unreliable, and it also prevents the drift that accumulates in sequential pairwise registration. We anchor the map in geographic coordinates instead of assembling it as a free-floating image, and export it as a georeferenced raster.

Figure 1: This is the result of our current algorithm for creating a terrain canvas

Ground station and payload release

We chose not to build a custom ground control station. The mandated judges' display is satisfied by a configured Mission Planner instance running flight-proven software, which freed our limited engineering hours for perception and geolocation work, where we had no prior flight-proven capability. Alongside it runs a supervisory application built on pymavlink. It receives target candidates over the data link and presents each one with its confidence, supporting observation count and image crop. The operator confirms the target before the release is armed. The competition awards no points for detection autonomy, while a misclassification costs 30 points per delivery, so a one-second human confirmation is the cheapest insurance available to us. The release point itself is computed onboard by a solver modelling the payload's ballistic trajectory and subsequent parachute drift, and software commands the release instead of the pilot, which removes human reaction time from the equation. Field trials have demonstrated roughly 5 m CEP against a 50 ft scoring radius.

How we test it

Verification is organised in four levels. Geolocation and the release solver are pure geometric computations, so we check them against synthetic scenes with known target positions. The same method lets us perturb individual inputs such as attitude, timestamp or terrain elevation and validate the error budget against the actual implementation instead of against theory. Mission logic, waypoint sequencing, failsafe behaviour and the release command path are exercised end-to-end in ArduPilot SITL simulation, which we are extending with a synthetic camera source so the perception pipeline can be tested in closed loop before flight hardware is available. On the bench, we re-evaluate the detection model after TensorRT export, validate mapping by replaying complete image sets from previous flights as a fixed regression fixture, and characterise the compute node's thermal and throughput behaviour under sustained concurrent load in its installed enclosure. Field campaigns at airfields around Kraków validate the full chain from image capture through detection, geolocation, operator confirmation and release. The Droniada 2026 competition doubled as a rehearsal of that operational timeline.