VAAS-X SDK · EDGE & AUTONOMOUS SYSTEMS

Persistent Mission Memory, With No Cloud in the Loop

How Bootstrap gives UAVs, robots, and autonomous vehicles operational memory that survives power cycles and disconnections — validated live on £30 microcontroller hardware, the same codebase used on an enterprise cluster.

£30	800M+	0	1
LIVE HARDWARE FLOOR	ESP32 UNITS DEPLOYED	CLOUD DEPENDENCY	CODEBASE, EVERY TIER

VAAS-X Ltd (SC873624) · July 2026 · vaasx.com

The problem: every mission starts from zero

UAVs, ground robots, and autonomous vehicles generate genuinely valuable operational data on every mission — what conditions preceded a good landing, what sensor pattern preceded a near-miss, what state the platform was in before a fault. Almost none of it survives shutdown. The next mission starts with no memory of the last one, and platforms operating without a connectivity guarantee can't lean on a cloud service to fill the gap.

Bootstrap runs entirely on-device

The `vaas-x` SDK's Bootstrap pipeline — statistical profiling, schema classification, episode accumulation, anomaly detection — executes locally on the platform's own hardware. There is no round-trip to the cloud in the ingestion or accumulation path, and no dependency on connectivity for the platform to keep building memory.

```
from vaasx import Bootstrap

brain = Bootstrap(api_key="YOUR_API_KEY", device_id="uav_07")
brain.connect("readings.jsonl")   # or a live sensor stream, ws(s), mqtt

# Works offline -- accumulates locally regardless of connectivity,
# syncs the resulting episodes when a link is available.
hits = brain.query("approach with crosswind gust", k=5)
```

The same `Bootstrap` class and API run identically whether the target is an enterprise EC2 cluster or the smallest embedded hardware the platform can carry — there is no separate "edge edition" to integrate against.

Validated: live on ESP32-class hardware

An M5Stack Cardputer — an ESP32-based microcontroller, roughly £30 in consumer hardware — was deployed as a live, fully offline memory shard. ESP32 itself has over 800 million units deployed globally across industrial, consumer, medical, and defence applications, making it a realistic floor for what "runs at the edge" needs to mean in practice, not a lab curiosity.

£30	800M+	0
HARDWARE VALIDATION POINT	ESP32 UNITS DEPLOYED GLOBALLY	CLOUD ROUND-TRIPS REQUIRED

The shard operated fully offline, accumulating numeric-channel episodic data on-device via Bootstrap's core pipeline and syncing to the main substrate in real time when connectivity was

available — confirming the deployment range from microcontroller to enterprise cluster runs on one codebase, not a scaled-down reimplementation. This validates Bootstrap itself at the hardware floor; it does not exercise `ArtificialCognition`'s heavier vision/audio/depth encoders, which need more compute than an ESP32-class chip provides — see the multimodal section below for what that path has and hasn't been validated against.

Multimodal sensor fusion (ArtificialCognition, Professional tier)

For platforms carrying more than numeric telemetry, `ArtificialCognition` fuses active sensor modalities — IMU/motion, vision, audio, depth, thermal — into the same episodic memory structure Bootstrap builds from numeric channels, so cross-modal recall works without a separate pipeline per sensor type.

```
from vaasx.ac import ArtificialCognition

ac = ArtificialCognition(brain)
fused = ac.encode(text="approach with crosswind gust")
episode_id = ac.ingest(fused, device_id="uav_07")
ac.outcome(episode_id, success=True)
# ac.retrieve(fused) -- not available yet; see Scope, honestly below
```

> **SCOPE, HONESTLY**
>
> Text encoding runs today and routes through the same production episode pipeline Bootstrap itself uses — ingest and outcome feedback on text-encoded episodes are real, not a mock. The vision, audio, depth, and thermal encoders construct and run structurally but wrap pretrained third-party models that have not been exercised end-to-end in this environment; the IMU encoder runs with no trained weights loaded, so its embeddings are not yet meaningful for retrieval. Cross-modal retrieve() itself is not available today — it depends on a server-side endpoint that hasn't shipped — so treat every modality except plain text ingestion as early access, not a working feature, until noted otherwise on the docs site. Numeric-channel telemetry through Bootstrap itself carries no such caveat — that path is fully production, as the CMAPSS and ESP32 validations above both show.

Get started

Free tier: instant API key, no credit card — `pip install vaas-x`, connect a source, query your first episode in under ten minutes.

Full guide: vaasx.com/docs/getting-started.html | Enterprise: hello@vaasx.com