FlyBot on AWS: system requirements

What the system has to do, how well, and what you need to try it, run it or build a trap camera. State of the system: 27 October 2026. Measured values come from the files named; "not measured" means nobody measured it.

1. Functional requirements

#	Requirement	How the system meets it	Where
F1	Count the fruit flies on every trap photo	YOLO26n on the OpenCV 5 DNN engine, NMS and fruit-fly box rules; 2 x 2 tiles on sticky cards	`lambda/vision/sentinel/vision/`, report section 4
F2	Refuse to count a bad photo	quality gate (blur, glare, dark, IR, missing trap), occlusion, moved camera, suspected person: "not counted" with the reason	`quality.py`, `motion.py`
F3	Separate new flies from old ones	registration with the previous photo (SIFT + FLANN + USAC MAGSAC) and box matching	`register.py`, `diff.py`
F4	Decide what should happen next	rule planner on every photo, LLM planner for questions, 14 typed tools, guardrails, a trace per photo	`lambda/vision/sentinel/agent/`
F5	Act only with a person's approval	every action is a proposal shown as a confirm card in FlyBot and on Telegram; only Confirm runs it; cards expire after 24 h	FlyBot `sentinelService.js`
F6	Close the loop	a confirmed capture produces a new photo that is analysed again	report section 8
F7	Two camera families	EZVIZ cloud cameras through the EZVIZ Open Platform; the open ESP32-S3 trap camera over HTTPS and IoT Core	report section 3.1
F8	Manage self-built cameras	zero-touch registration, QR claim, shadow and RPC control, KVS WebRTC live view, OTA updates on release channels, remote log, crash reports	`docs/CONTRACTS.md` sections 2-3
F9	Answer growers' questions	FlyBot's chatbot calls the Lambda in deep mode; answers from OpenCV's numbers, never from the pixels	report section 8
F10	Alerts and reports	thresholds, surges and a trap camera's critical battery by Telegram, email and web push; PDF reports	FlyBot
F11	Let judges try it without an account	Guest access: shared read-only account for two hours; a labelled simulated farm network	report section 3.2
F12	Vietnamese and English	the app, the chatbot, proposal reasons and alerts in the reader's language	FlyBot locales, `reasonI18n`

2. Quality requirements

Quality	Requirement	Status (source)
Accuracy	detector good enough on public trap pictures	test mAP@50 0.953, F1 0.9006 through the OpenCV path (`models/MODEL_CARD.md`); field accuracy against human counts **not measured**; agent decisions judged by the agronomy advisor: 43 of 49 correct, 87.8 % (95 % interval 75.8-94.3 %), 1 unsure (`eval/results/expert_review.json`); after the lure-floor glare limit, 1 of the 7 rejected or unsure recaptures remains on the same photos and none of the 43 correct ones changed (`eval/results/glare_rule_ab.json`)
Safety	no camera change without a person	38/38 failure-injection runs, 0 false actuations, earlier build with fake devices (`eval/RESULTS.md` section 4)
Latency	one photo answered within seconds	Lambda warm p50 980 ms, cold 3,633 ms (`bench/results/lambda_latency.json`)
Cost	cheap enough for a small farm	about 17 USD a month for the demo, about 0.046 USD per 1,000 photos (`infra/COST.md`)

Quality	Requirement	Status (source)
Privacy	no person identified from a trap photo	no photo to an LLM, face regions blurred, the Lambda stores no picture (`docs/RESPONSIBLE_USE.md`)
Security	no secret in git or on disk; least privilege	Secrets Manager, scoped roles, one certificate per camera, 15-minute viewer credentials (report section 5)
Observability	every decision explainable	a trace per photo (planner, decisions, tool calls, timings, fallback) stored by FlyBot; CloudWatch alarms
Reproducibility	a judge can rebuild and rerun	pinned dependencies, `make test`, `make replay-sample`, one-command `./deploy.sh`
Availability	demo online for the judging period	one region, one server, no high availability (report section 9)

3. To try it

- A web browser (Chrome, Edge, Firefox or Safari, desktop or phone). Open https://flybot.agrhub.com/auth/login and press **Guest access**. No account, no email.

4. To run it locally (no AWS account, no hardware)

Need	Version
OS	Linux or macOS with `make`; Windows works without `make` (README "Run locally")
uv, Python	uv, Python 3.12 (`lambda/vision`)
OpenCV	`opencv-python-headless==5.0.0.93` (pinned in `lambda/vision/requirements-lambda.txt`; 4.x is refused)
Other Python packages	`numpy==2.5.4`, `pydantic==2.14.1`, `anthropic==1.14.0` (pinned in the same file)
Models	`make models` (YOLOv7-tiny baseline, SHA-256 checked); YOLO26n from the public model bucket (README)
Optional	`ANTHROPIC_API_KEY` for LLM answers in deep mode; Docker Compose to run FlyBot itself locally

5. To deploy it on AWS

Need	Detail
AWS account	rights to create IAM roles; region ap-southeast-2; quotas for one Elastic IP and the instance vCPUs (checked by `./deploy.sh` preflight)
Tools on the operator's computer	AWS CLI v2, SAM CLI, uv or Python 3.9+, Node.js 22.12+ with npm, git, curl; no Docker
FlyBot checkout	the `flybot/` submodule or https://github.com/agrhub/flybot, branch `feat/aws`
Settings	`deploy.env` from `deploy.env.example` (domain, alarm email, optional EZVIZ, Telegram, SMTP, Vertex AI, Anthropic keys)
DNS	an A record for the domain to the Elastic IP the script prints
Optional	a GitHub token to publish FlyBot releases (otherwise the private artifacts bucket); an external MongoDB or S3-compatible store

What the stack creates and runs:

Resource	Size or version
Vision Lambda	container image, arm64, Python 3.12, 2,048 MB, 120 s timeout, one synchronous call per photo
Server	EC2 t4g.micro (Graviton), Amazon Linux 2023 arm64, Elastic IP, Node 22, MongoDB 7, Caddy 2.10.2 with automatic TLS
Cameras	IoT Core thing type `camos-camera`, device policy, shadow, rule; one Kinesis Video Streams signaling channel per CamOS camera

Resource	Size or version
Storage and secrets	S3 buckets (photos, artifacts), ECR, Secrets Manager
Monitoring	CloudWatch logs and alarms, SNS email
Network	HTTPS 443 and HTTP 80 (ACME) to the server; SSH 22 only from the owner's IP and optional; cameras reach the server over HTTPS and IoT Core over MQTT with mutual TLS (8883)

6. To build a trap camera

Part	Detail
Board	ESP32-S3 N16R8 CAM board (16 MB flash, 8 MB PSRAM) with an OV5640 wide-angle camera, or the Freenove ESP32-S3 CAM (FNK0085); microSD, USB-C (CH340)
Wi-Fi camera	the board and a trap housing: about US$19 in all with the US$3.00 hanging solar trap (419,640 VND of parts; the trap is US$9.90 on Amazon); a microSD card is optional
Field camera	as built in the hanging trap: plus a 4G module (A7680C) and a GPS module (ATGM336H), both with their antennas, a flat LiPo cell (103450, 2,000 mAh), a TP4056 charger and a 5 V boost board: about US$39 in all with the housing (944,760 VND of parts at 26,050 VND/USD, Vietcombank, 21 Oct 2026, plus the US$3.00 housing); optional microSD, solar panel and active GPS antenna +US$9.67; the reference build in an IP65 box: about US$60 (prices read on 17, 24 and 25 Oct 2026, camos-esp32-trap docs/BOM.md)
Wiring	camos-esp32-trap docs/WIRING.md (reference design); the bench trap as built: `4G-trap-wiring.pdf` in the Devpost package
Firmware	camos-esp32-trap v0.9.2; first install with the web flasher in Chrome or Edge, OTA from FlyBot afterwards; built with ESP-IDF v5.4.4
Network	Wi-Fi 2.4 GHz, or a nano SIM with data for 4G
Power	USB 5 V, or the LiPo cell with solar charging; battery life and solar charging **not measured**