FASS SmartPark-IoT - Developer Implementation Spec (AC)

R&D; and implementation handoff document for the engineering team.

Site: Sabanci University Tuzla Campus - FASS parking lot

Edge HW: Raspberry Pi 4, coverage: all visible slots

Power profile: AC (see Section 3)

Date: 2026-01-03

Goal: Build a deployable IoT sensing node that publishes per-slot occupancy as event-driven telemetry (MQTT), with reliability, operations monitoring, and remote configuration. ML is optional and must not compromise IoT objectives (bandwidth, uptime, maintainability, privacy).

1. Scope and non-scope

Scope is limited to the FASS parking lot. The system must cover all parking slots visible from the selected camera viewpoint.

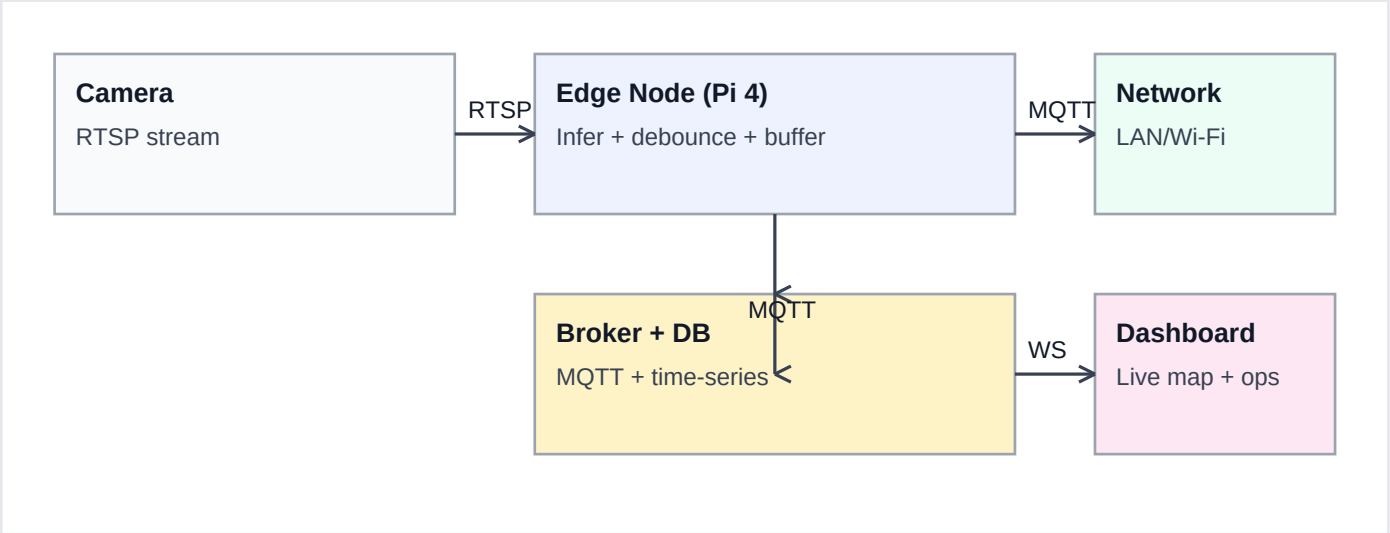
• In-scope: on-edge sensing pipeline, calibrated slot mapping, MQTT telemetry, buffering/replay, dashboard integration, field deployment.

• Out-of-scope: mobile app, reservations/payment, campus-wide multi-lot rollout (unless time permits), identity/plate tracking.

• Privacy requirement: do not stream or store raw video off-device by default; publish only structured occupancy telemetry.

2. System architecture (IoT-first)

Implement the system as a set of services with clear responsibilities. Prefer robust ops over model complexity.



2.1 Services (edge)

• capture: RTSP ingest, frame timestamping, exposure notes, drop detection.

• calibration: ROI mask + slot polygons loading; optional homography rectification.

• occupancy: per-slot scoring, debounce/hysteresis, UNKNOWN gating, event generation.

• telemetry: MQTT publish, QoS handling, store-and-forward queue, replay.

• health: periodic node metrics (fps/temp/mem/net/buffer depth) + camera stream status.

• config: subscribe to config topic, validate and apply updates safely (rollback support).

2.2 Server components

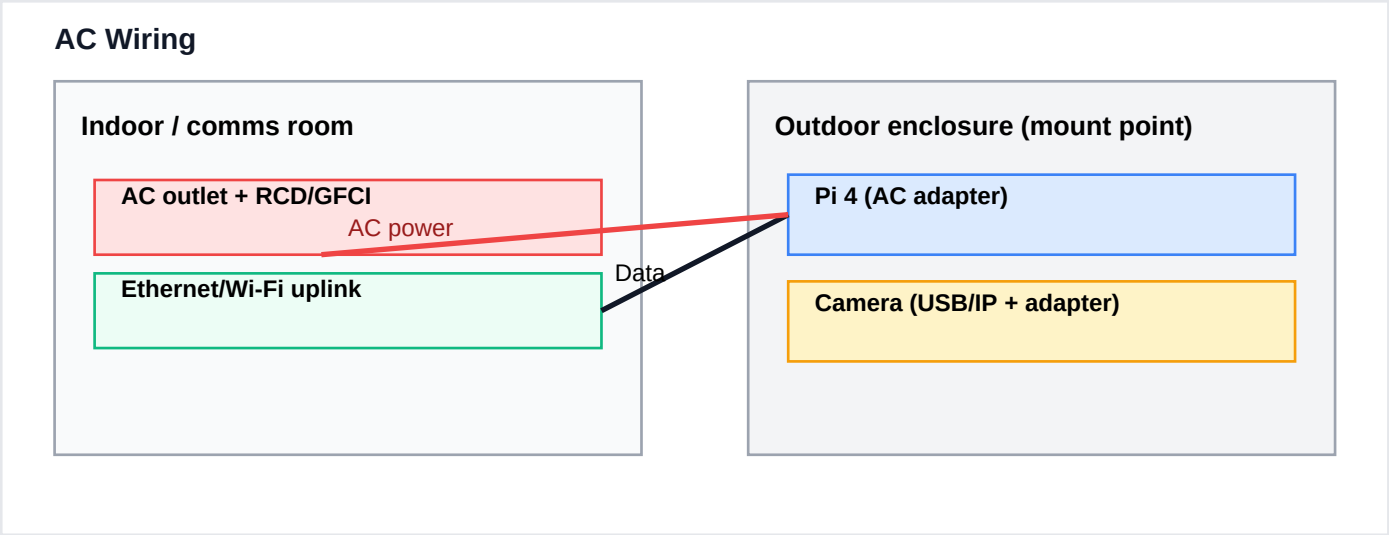
• MQTT broker with ACL (per-node credentials).

• DB for events and telemetry (time-series DB preferred).

• Dashboard: live lot state, history, ops panel, basic alerts (node offline, buffer growing).

3. Power and wiring plan

Follow the wiring diagram and safety/robustness notes. This is the main divergence between the PoE and AC versions.



3.1 AC acceptance criteria

• Use RCD/GFCI protection and outdoor-rated AC routing per campus rules.

• Convert AC to DC near enclosure; avoid long USB power runs.

• Pi and camera powered for 8+ hours without undervoltage; enclosure sealing verified.

• Document safety checklist and maintenance access plan.

4. Interfaces and data contracts (must implement)

MQTT is the primary interface between edge and server. Keep payloads stable; version artifacts.

Topic	QoS	Direction	Publish rate	Required fields
su/parking/fass/slot/<slot_id>/state	1	Edge->Broker	On change	slot_id, occupied|unknown, confidence, ts_utc, dwel
su/parking/fass/summary	0	Edge->Broker	Every 5-10 s	free_count, occupied_count, unknown_count, ts_utc
su/parking/fass/node_health	0	Edge->Broker	Every 10-30 s	uptime_s, fps, cpu_temp_c, mem_mb, dropped_fra
su/parking/fass/config	1	Broker->Edge	On demand	publish_period_s, debounce_s, enter_thr, exit_thr, r

4.1 JSON payload rules

• Include ts_utc in ISO-8601 UTC for all messages.

• Include roi_version and model_version in state and summary.

• State events are only published on transitions after debounce confirms stability.

• Unknown is a valid state when confidence is below threshold; avoid forced guesses.

5. Calibration artifacts and tooling

• roi_mask.png: binary mask of parking region.

• fass_slots_v1.json: polygons for all visible slots (slot_id + list of points).

• homography_v1.json: optional, recommended if camera is angled.

• overlay_check tool: shows live video with ROI/slot overlays to validate calibration.

5.1 Slot polygon JSON schema (example)

{ "roi_version": "v1", "image_size": [1920, 1080], "slots": [{"slot_id": "FASS_001", "poly": [[x1,y1],[x2,y2],[x3,y3],[x4,y4]]}, ...] }

6. Occupancy engine (IoT behavior)

Implement debouncing and hysteresis as the default behavior. This is required for stable IoT sensing.

• Per-slot score in [0,1].

• Enter threshold != exit threshold (hysteresis).

• Debounce: require state to hold for K seconds before transition.

• Emit dwell_s in events (time spent in previous state).

• Confidence gating: if score is between thresholds or noisy, set state UNKNOWN.

6.1 Minimal baseline (no-ML) must exist

• Background difference within ROI + threshold features.

• Used for regression tests and as fallback if model fails.

7. Reliability and operations (must-have)

• Store-and-forward queue (SQLite): append on publish failure; replay on reconnect; dedupe on (slot_id, transition_ts).

• Systemd services with Restart=always; log rotation.

• Heartbeat metric and alert when node is offline > 30 s.

• Camera reconnect strategy; expose stream_up boolean in health telemetry.

7.1 Test cases (engineer must run)

• Network outage test: pull uplink for 2 minutes; verify no lost transitions and correct replay ordering.

• Camera outage test: stop RTSP stream; verify reconnect behavior; health shows stream_down.

• Thermal soak: run 1 hour; ensure fps stable and no throttling.

8. Milestones and deliverables

• D1: Site survey + mount plan + sample clip from final viewpoint.

• D2: Calibration v1 (all visible slots) + live overlay validation.

• D3: Edge pipeline stable for 60 min + local overlay + basic metrics.

• D4: MQTT telemetry live (state + summary + health) + server storage.

• D5: Dashboard live map + history + ops panel.

• D6: Reliability features + outage tests + evaluation report.

9. Implementation backlog (epics)

Epic	Key tasks	Acceptance criteria (done means)
E0.5 AC power integra[illegible]	Define AC routing; RCD/GFCI; PSUs; cable gland[illegible]	[illegible] safety checklist [illegible] undervolta
E0 Site survey & moun[illegible]	Choose mount point; confirm FOV covers all visib[illegible]	[illegible]
E1 Calibration artifacts	Create roi_mask; label all slot polygons; export fa[illegible]	[illegible]; ROI versioned; J
E2 Edge capture pipel[illegible]	RTSP ingest; frame timestamping (UTC); exposur[illegible]	[illegible] reported; fps withi

E3 Occupancy engine	ROI crop/rectify; occupancy score; debounce + hysteresis; UNKNOWN behavior	False toggles low; slot state event generation delay measured
E4 Telemetry & schemas	MQTT client; topics; JSON payloads; QoS selection; retained messages	Broker receives events, summary, health; messages in
E5 Reliability & ops	Store-and-forward queue (SQLite); replay; heartbeat; watchdog; structured logs	Network outage test passes; no lost events; node_health recovery; health n
E6 Dashboard	Live map view (slot colors); free/occupied counts; time series; node health panel; alerts	Shows live state, snapshots, historical queries; indicates node
E7 Security hardening	MQTT auth + ACL; TLS if feasible; SSH key-only; firewall; least privilege	Unauthorized publish/subscribe; credentials stored safely; min

10. Repository and dev environment

Prefer Python for rapid iteration on Pi. Containerize server if possible (Docker Compose).

```
repo/ edge/ services/ (capture.py, occupancy.py, telemetry.py, health.py, config.py)
calibration/ (fass_slots_v1.json, roi_mask.png, homography_v1.json) tests/ (schema
tests, replay tests) systemd/ (unit files) server/ compose.yaml (mqtt broker, db,
dashboard) mqtt/ (acl, users) dashboard/ docs/
```

10.1 Suggested technologies

• Edge: Python 3.11+, OpenCV, paho-mqtt, SQLite, psutil.

• Server: Mosquitto (MQTT), InfluxDB or Postgres, Grafana or a minimal web dashboard.

• Transport: MQTT with QoS 1 for events, QoS 0 for summaries/health.

11. Definition of Done (project acceptance)

• All visible slots are mapped and reported with stable state (no flicker) under normal daytime conditions.

• Event-driven telemetry reduces bandwidth (no raw video off-device; state events only on changes).

• Reliability: network outage test passes with buffered replay and no lost events.

• Ops: dashboard shows node health; node restarts on failure automatically.

• Power: AC-powered node + camera sustained for 8+ hours with safety checklist and stable supplies.