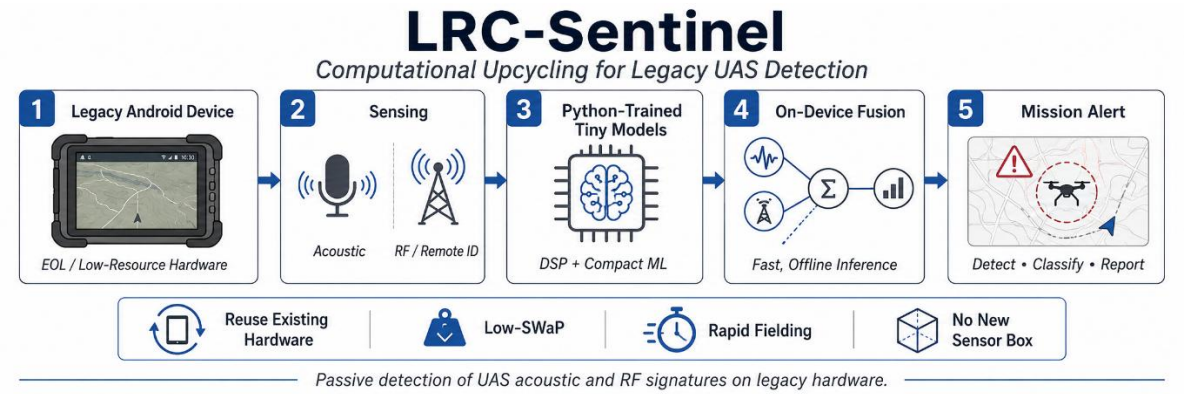
LRC-Sentinel: Computational Upcycling of Legacy Android Hardware for Passive UAS Acoustic and RF Detection



1. Identification and Significance of the Problem

The Department of War has invested heavily in ruggedized mobile, embedded, and communications hardware that remains fielded, familiar to operators, environmentally tested, and logistically understood. Much of this hardware is no longer considered capable of supporting modern autonomy, sensor fusion, or machine-learning workloads because it has limited RAM, CPU, storage, battery headroom, and operating-system support. The conventional modernization answer is to replace the device or add a new edge-compute box. That approach is slow, expensive, and often mismatched to tactical reality.

Groundbreaker Solutions proposes LRC-Sentinel, a Phase I prototype that demonstrates a different path: computationally upcycling a legacy Android rugged tablet or handheld into a passive small unmanned aerial system detection node. The system will use Python-trained signal-processing and machine-learning models, exported into compact mobile artifacts, to detect UAS acoustic and RF indicators on low-resource hardware. The Phase I demonstration will show that a legacy device can perform useful UAS detection within its native resource envelope, rather than requiring replacement hardware.

The proposed technical thesis is simple: many legacy devices contain enough sensing, compute, and communication capability to support narrowly scoped mission functions when the software is redesigned around their constraints. The innovation is not a new drone detector, a new radio, or a new processor. The innovation is a resource-aware semantic overlay that adds a modern detection function to existing hardware.

The initial target class will be legacy rugged Android devices such as the Samsung Galaxy Tab Active2 and Zebra TC51-class handheld. These devices are attractive Phase I targets because they are obtainable, representative of fielded enterprise/rugged mobile hardware, and constrained enough to make the demonstration meaningful. A representative low-resource device with 16 GB internal storage can be compared against a modern rugged Android baseline with 128 GB storage, creating a clear under-25% resource argument in at least one resource dimension. Phase I will document this comparison quantitatively and will also measure CPU, RAM, battery, thermal behavior, operating-system limits, microphone behavior, Wi-Fi/Bluetooth scan behavior, and model footprint.

LRC-Sentinel will demonstrate a multi-stage detection pipeline. Acoustic sensing will serve as the low-cost always-on trigger. RF and Remote ID observations will serve as corroborating evidence when available. The system will fuse these signals locally and emit a compact alert event that can be logged, displayed locally, or exported through a TAK/ATAK-compatible adapter during demonstration. ATAK integration is not the core innovation. It is a transition-facing interface showing how upcycled legacy nodes can contribute tactical observations to existing situational-awareness workflows.

The Phase I result will be a measured prototype, not a paper architecture. Success will be shown by running the detection pipeline on the selected legacy device, documenting latency, memory, CPU utilization, power draw, false alarm behavior, model size, and detection performance. The demonstration will establish whether the approach can provide useful UAS warning capability years earlier than a traditional hardware replacement path.

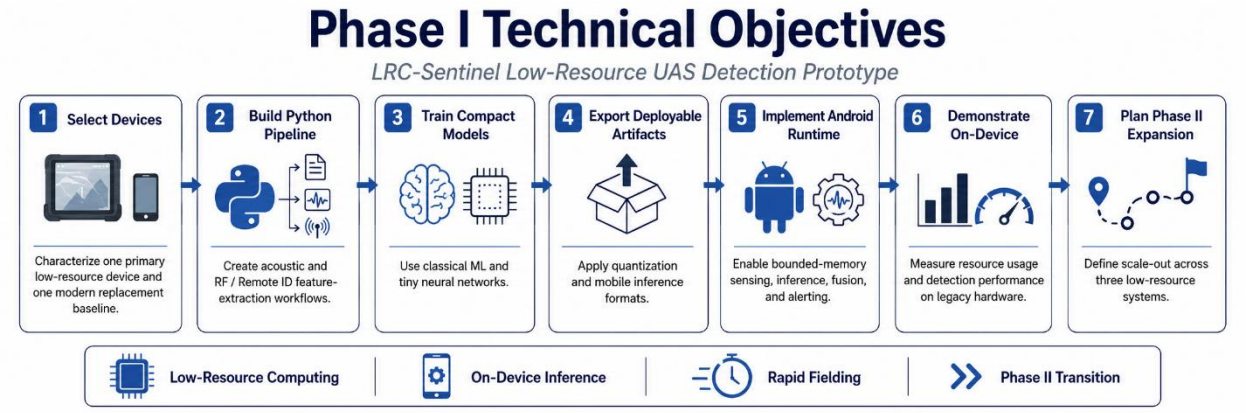
2. Technical Objective and Phase I Hypothesis

The Phase I objective is to prove that a low-resource legacy Android platform can run a modern passive UAS detection workload using resource-aware signal processing, compact machine learning, and opportunistic RF corroboration.

The central hypothesis is:

A legacy Android device that is widely believed unsuitable for modern AI-enabled sensing can detect UAS acoustic signatures and corroborating RF/Remote ID indicators when the workload is redesigned as a staged, compressed, and duty-cycled semantic overlay.

The Phase I prototype will not attempt to build a full counter-UAS system. It will demonstrate feasibility of a specific, measurable capability: passive UAS detection and alerting on constrained legacy hardware.



The core technical question is not whether drone detection can be performed on powerful hardware. It can. The core question is whether a useful subset of that capability can be made to run on hardware with severe RAM, storage, CPU, OS, and battery constraints. LRC-Sentinel attacks that question directly.

3. Proposed Innovation

The proposed innovation is a computational-upcycling overlay that adds UAS detection capability to existing low-resource devices without requiring hardware redesign. The overlay has five technical elements.

First, the platform characterization layer measures actual device constraints. It records CPU class, core behavior, RAM availability, storage, battery drain, audio capture behavior, operating-system version, microphone sampling limits, Wi-Fi/Bluetooth scan constraints, and thermal behavior. This establishes the low-resource case and prevents overclaiming.

Second, the acoustic front end uses the device microphone as the primary sensor. The runtime maintains a bounded audio ring buffer, performs low-cost signal conditioning, and extracts compact features such as log-Mel summaries, MFCCs, RMS energy, onset strength, harmonic ratios, and pitch-related statistics. The system avoids storing long raw audio histories. It uses fixed-size buffers and staged inference to remain inside the device resource envelope.

Third, the RF/Remote ID front end collects available RF evidence without assuming unrealistic access to raw radio firmware. On stock Android, the system can observe Bluetooth Low Energy advertisements, Wi-Fi scan metadata, signal strength trends, and Remote ID messages when device support and permissions allow. Where appropriate, the Phase I design may also support an optional COTS RF front end or SDR-derived spectral snapshots, but the main compute and fusion host remains the legacy Android device. This is consistent with the low-resource computing objective because the approach reuses the existing device and integrates low-SWaP COTS components only where necessary.

Fourth, the model pipeline is designed for constrained deployment from the start. Python is used for training, experimentation, feature analysis, dataset construction, and model compression. Deployment artifacts are kept small. Classical models such as logistic regression, linear SVM, and random forest provide transparent baselines. Tiny

neural models, including small 1D or 2D convolutional networks, provide a higher-performance path when the resource budget allows. Quantization, pruning, and knowledge distillation will be evaluated to reduce model size and latency.

Fifth, the fusion and alerting layer converts noisy sensor evidence into an operator-usable event. Acoustic detections are temporally smoothed. RF and Remote ID observations raise confidence when present. The system avoids alert spam through hysteresis and event update logic. The output can be logged locally, displayed in the prototype application, or emitted through a TAK/ATAK-compatible event adapter.

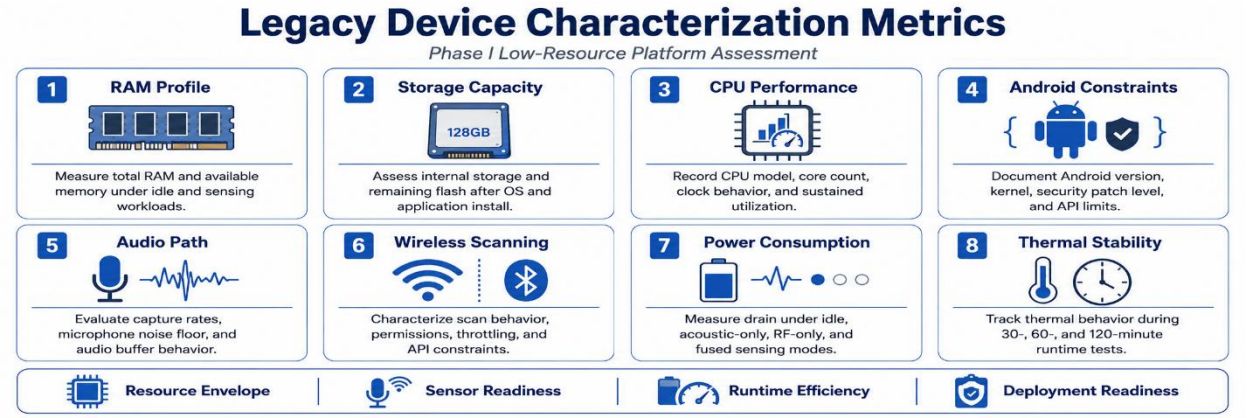
The novelty is the integrated resource discipline. LRC-Sentinel is not a generic Android app with an AI model attached. It is a staged sensing pipeline designed around legacy constraints: cheap trigger first, compact classifier second, RF corroboration only when useful, bounded memory everywhere, and measured device performance as a primary deliverable.

4. Technical Approach

4.1 Device Selection and Low-Resource Analysis

Phase I will begin by selecting the primary legacy device and defining the proposed new-system baseline. The current recommended primary target is a Samsung Galaxy Tab Active2-class rugged tablet or Zebra TC51-class rugged handheld. The modern baseline will be a current rugged Android device or dedicated edge sensor class with materially greater storage, RAM, CPU, OS support, and battery headroom.

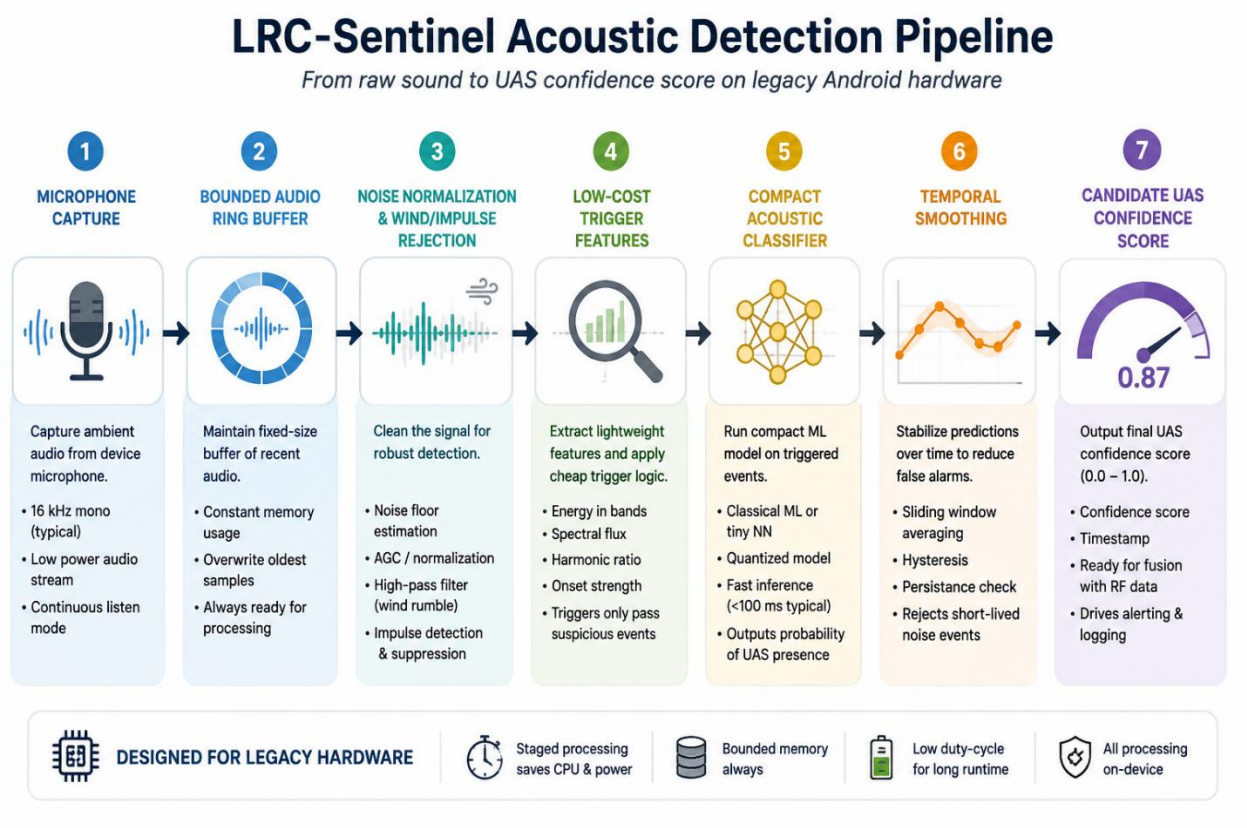
The low-resource case will be documented in the Month 1 report and refined by Month 3. At minimum, the analysis will include:



The proposal will argue low-resource status through measured data. For example, a 16 GB legacy storage device compared to a 128 GB modern rugged baseline represents 12.5% of the proposed new-system storage resource, satisfying the under-25% criterion in one resource dimension. Similar comparisons will be made where RAM, CPU, or operating-system limits provide a stronger argument.

4.2 Acoustic Detection Pipeline

The acoustic pipeline will treat the legacy Android microphone as a passive UAS sensor. The target signature is not a visual drone image. It is the rotor, motor, propeller, and motion-induced acoustic pattern produced by small UAS platforms. These signatures may include blade-passing tones, harmonic structures, broadband motor noise, and temporal motion patterns.



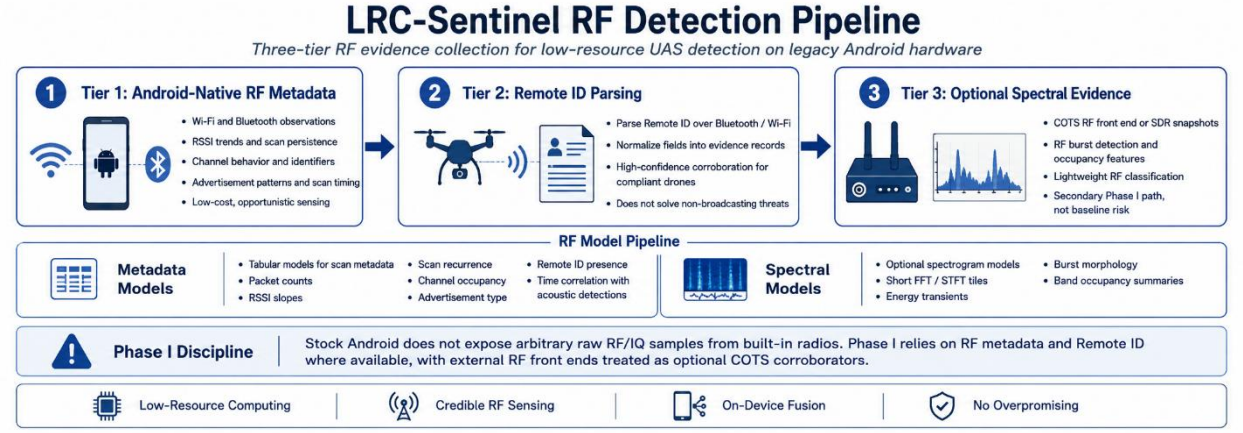
The Python training pipeline will produce the acoustic models. Raw or collected audio will be normalized, downsampled for model input, segmented into short windows, and transformed into compact feature representations. Candidate features include MFCCs, log-Mel spectrogram summaries, RMS energy, spectral flux, onset strength, harmonic ratio, and pitch statistics. These features are intentionally chosen because they are inexpensive to compute and can support both classical ML and tiny neural models.

The first model family will be classical ML. Logistic regression and linear SVM baselines will be trained on pooled feature vectors. These models are small, explainable, fast, and appropriate for legacy deployment. A random forest or gradient-boosted tree model may be evaluated as an intermediate robustness option, but only if its memory and latency remain acceptable.

The second model family will be compact neural models. Candidate models include small 1D convolutional networks over short feature sequences and small 2D convolutional networks over log-Mel or STFT tiles. These models will be quantized to 8-bit artifacts where possible. The goal is not maximum accuracy; the goal is the best accuracy/resource tradeoff that can run continuously or near-continuously on the target device.

The system will use staged execution. A cheap signal gate will run continuously. The classifier will run only when the signal gate detects acoustic structure consistent with a candidate UAS event. This reduces CPU load and battery drain while keeping alert latency bounded.

4.3 RF and Remote ID Detection Pipeline



The RF pipeline will be built in three tiers.

Tier 1 is Android-native RF metadata. The system will collect available Wi-Fi and Bluetooth observations such as RSSI trends, scan persistence, channel behavior, device identifiers, advertisement patterns, and scan timing. This tier is low cost and can run opportunistically.

Tier 2 is Remote ID parsing. When Remote ID messages are available through Bluetooth or Wi-Fi paths supported by the target device, the system will parse and normalize the message fields into a structured evidence record. Remote ID evidence can provide high-confidence corroboration for compliant drones, but the proposal will not claim that Remote ID solves non-compliant or non-broadcasting UAS threats.

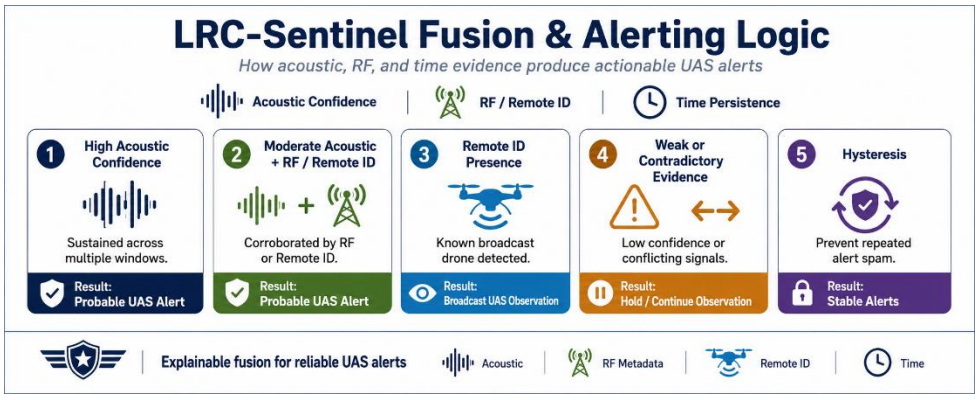
Tier 3 is optional spectral evidence from a COTS RF front end or SDR-derived snapshots. This is a secondary Phase I path, not the primary risk. It can support RF burst detection, spectrum occupancy features, and lightweight RF classification, but it will not be required for the baseline demonstration unless device-native RF access proves insufficient for useful corroboration.

The RF model pipeline will include simple tabular models for scan metadata and optional spectrogram models for RF snapshots. Metadata features may include packet counts, RSSI slopes, scan recurrence, channel occupancy, advertisement type, Remote ID presence, and time correlation with acoustic detections. Spectral features may include short FFT/STFT tiles, energy transients, burst morphology, and band occupancy summaries.

The key proposal discipline is honesty about Android RF constraints. Stock Android does not expose arbitrary raw RF/IQ samples from the built-in radios. Therefore, the Phase I approach uses RF metadata and Remote ID where available, and treats external RF front ends as optional COTS corroborators. This makes the technical claim credible and avoids overpromising.

4.4 Fusion and Alerting

The fusion layer will combine acoustic confidence, RF metadata, Remote ID evidence, and temporal persistence into a single alert state. The initial logic will be intentionally simple and explainable:



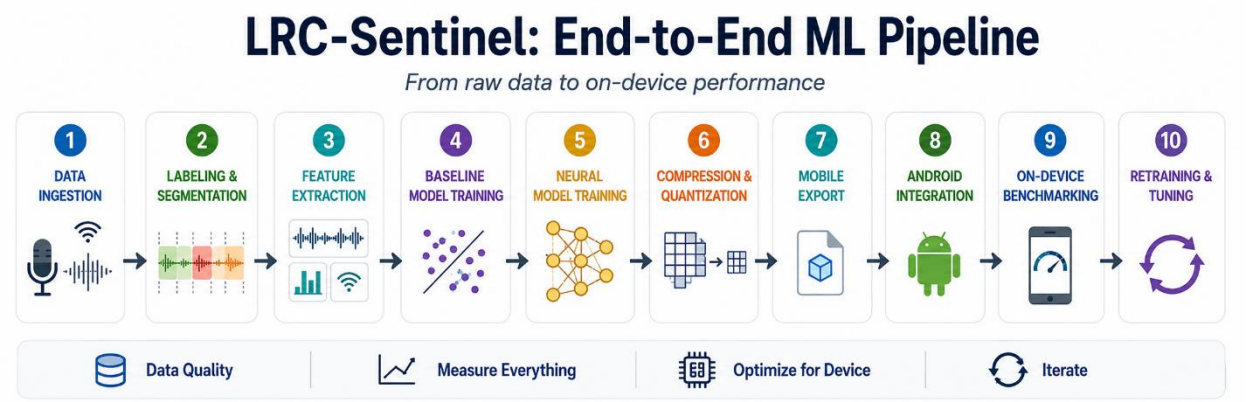
The fusion layer will preserve evidence for review. Each alert will include timestamp, device ID, confidence, modality contributions, acoustic score, RF/Remote ID score, and resource state. Where permitted, the prototype may retain a short pre-alert and post-alert feature window, but it will avoid persistent raw audio collection unless explicitly enabled for test purposes.

The demonstration output will include local logs, an on-device dashboard, and an optional TAK/ATAK-compatible event adapter. The adapter will show how a computationally upcycled device can contribute to tactical situational awareness without replacing existing hardware.

5. Model Development in Python and Deployment to Legacy Hardware

Python will be used as the model-development environment because it provides mature tooling for digital signal processing, scientific computing, classical ML, neural-network training, dataset management, and model compression. The deployed legacy device will not run the full Python stack. Instead, Python produces compact artifacts and validated preprocessing logic that are reimplemented or exported into the Android runtime.

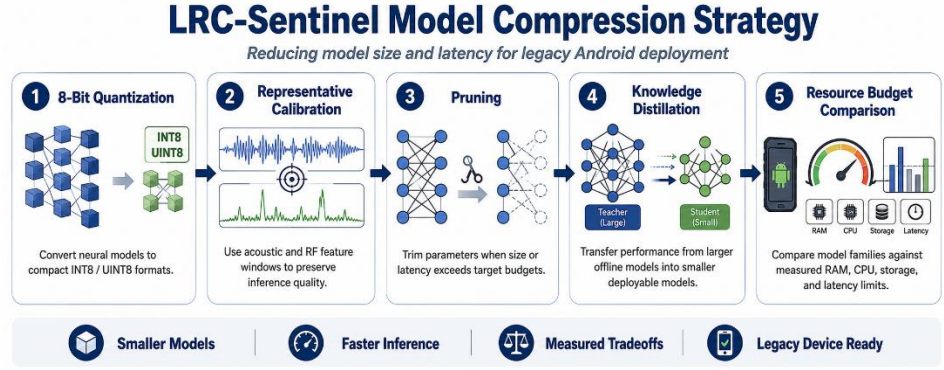
The development pipeline will be:



For acoustic models, Python will generate reproducible datasets from public UAS audio, collected local drone audio, and negative background sounds such as traffic, HVAC, wind, speech, birds, lawn equipment, generators, and quiet environments. Synthetic augmentation will mix drone sounds with background noise at controlled signal-to-noise ratios. Device simulation will add resampling, clipping, band-limiting, gain changes, and microphone noise to reduce the gap between training data and legacy Android microphone capture.

For RF models, Python will process Wi-Fi/Bluetooth/Remote ID logs and any RF spectral snapshots collected during Phase I. The RF model will not depend on classified waveforms or inaccessible radio internals. It will use available features and open test data where appropriate.

The first deployed model will likely be a classical ML classifier over pooled acoustic features. This gives a low-risk path to an on-device demonstration because the model artifact is tiny and inference is fast. A second deployed model will be a quantized tiny neural model if benchmarking shows it fits the resource envelope. The final prototype may use both: a classical trigger model for always-on sensing and a quantized neural classifier for triggered confirmation.



Deployment artifacts may include TensorFlow Lite, LiteRT, ONNX Runtime Mobile, or a hand-coded classical model runtime depending on model family and target device behavior. The Android application will use Java/Kotlin for lifecycle, permissions, scanning, logging, and UI. Native C/C++ through the Android NDK may be used for tight loops, FFT, feature extraction, or inference where it materially improves latency or memory use.

The hot path will avoid repeated allocation, avoid unbounded queues, avoid retaining long histories, and avoid unnecessary Java/native crossings. The runtime will use fixed-size buffers, preallocated arrays, and duty-cycled sensing. This is essential to proving that the capability can coexist with the constraints of the legacy platform.

6. Phase I Work Plan and Milestones

The work plan follows the required Phase I milestone structure.

Month 1: Architecture, Platform Selection, and Functionality Specification

Groundbreaker Solutions will select the primary low-resource device and define the new-system comparison baseline. The team will document candidate devices, acquisition path, operating-system constraints, sensor interfaces, and initial resource assumptions. The Month 1 report will include the initial LRC-Sentinel architecture, functional requirements, evaluation metrics, data plan, model-development plan, and prototype demonstration concept.

Deliverables:

- Initial architecture report.
- Selected platform and baseline comparison.
- New functionality specification.
- Initial risk register.
- Test and evaluation plan draft.

Month 2: Dataset Assembly and Runtime Skeleton

The team will assemble the initial acoustic and RF datasets. This will include public data, locally collected negative background data, controlled drone audio if available, synthetic augmentation, Wi-Fi/Bluetooth scan captures, and Remote ID test captures or simulated messages. In parallel, the team will build the Android runtime skeleton, including audio capture, bounded buffering, logging, device profiling, and basic scan collection.

Deliverables:

- Initial dataset inventory.
- Android capture and logging prototype.
- Python feature-extraction notebooks or scripts.
- Initial device profiling results.

Month 3: Platform Acquisition, Metrics, and Initial Analysis

The team will complete acquisition and configuration of the selected platform. Baseline resource measurements will be collected. Initial classical ML models will be trained and evaluated offline. The Month 3 report will document acquisition, evaluation metrics, early analysis, low-resource calculations, and initial model results.

Deliverables:

- Month 3 metrics and analysis report.
- Device characterization package.
- Initial acoustic classifier.
- Initial RF/Remote ID evidence collector.
- Updated feasibility and risk assessment.

Month 4: Interim Prototype

The team will integrate the acoustic feature pipeline, first on-device model, RF/Remote ID evidence collector, fusion logic, and local alerting. The interim prototype will run on the selected legacy device. Performance will be measured under controlled test conditions.

Deliverables:

- Interim prototype demonstration build.
- On-device performance report.
- Model size, latency, RAM, CPU, and power measurements.
- Updated model compression results.
- Gap analysis for Month 5 demonstration.

Month 5: Prototype Demonstration

The team will demonstrate LRC-Sentinel running on the low-resource platform. The demonstration will show acoustic sensing, RF/Remote ID corroboration where available, fused alerting, local logging, and optional TAK/ATAK-compatible event output. The demonstration will document initial performance metrics and compare them against the proposed baseline.

Deliverables:

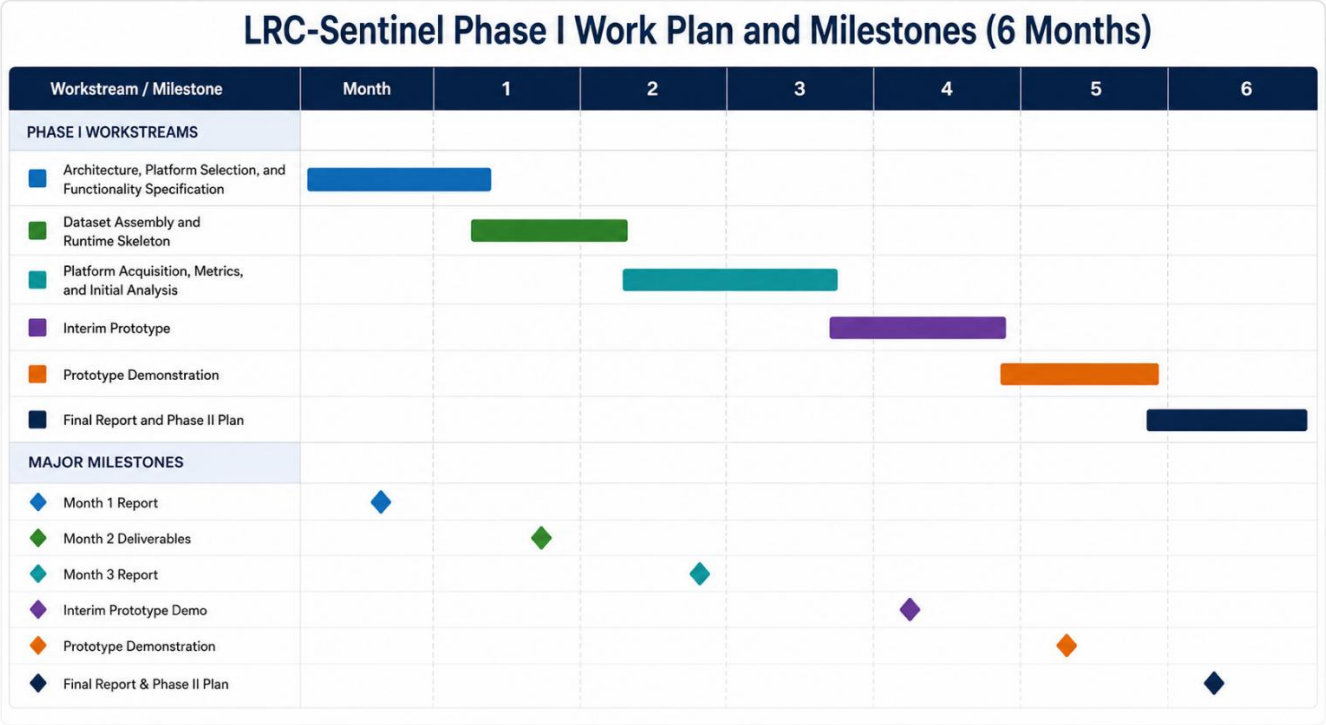
- Prototype demonstration.
- Demonstration dataset or test stream.
- On-device benchmark results.
- Alert logs and resource traces.
- Initial comparison against alternative modernization path.

Month 6: Final Report and Phase II Plan

The final report will summarize the approach, prototype architecture, algorithms, training process, model artifacts, resource measurements, evaluation design, performance results, limitations, and Phase II plan. The Phase II plan will identify at least three low-resource systems for expansion and define how LRC-Sentinel generalizes beyond the first Android surrogate.

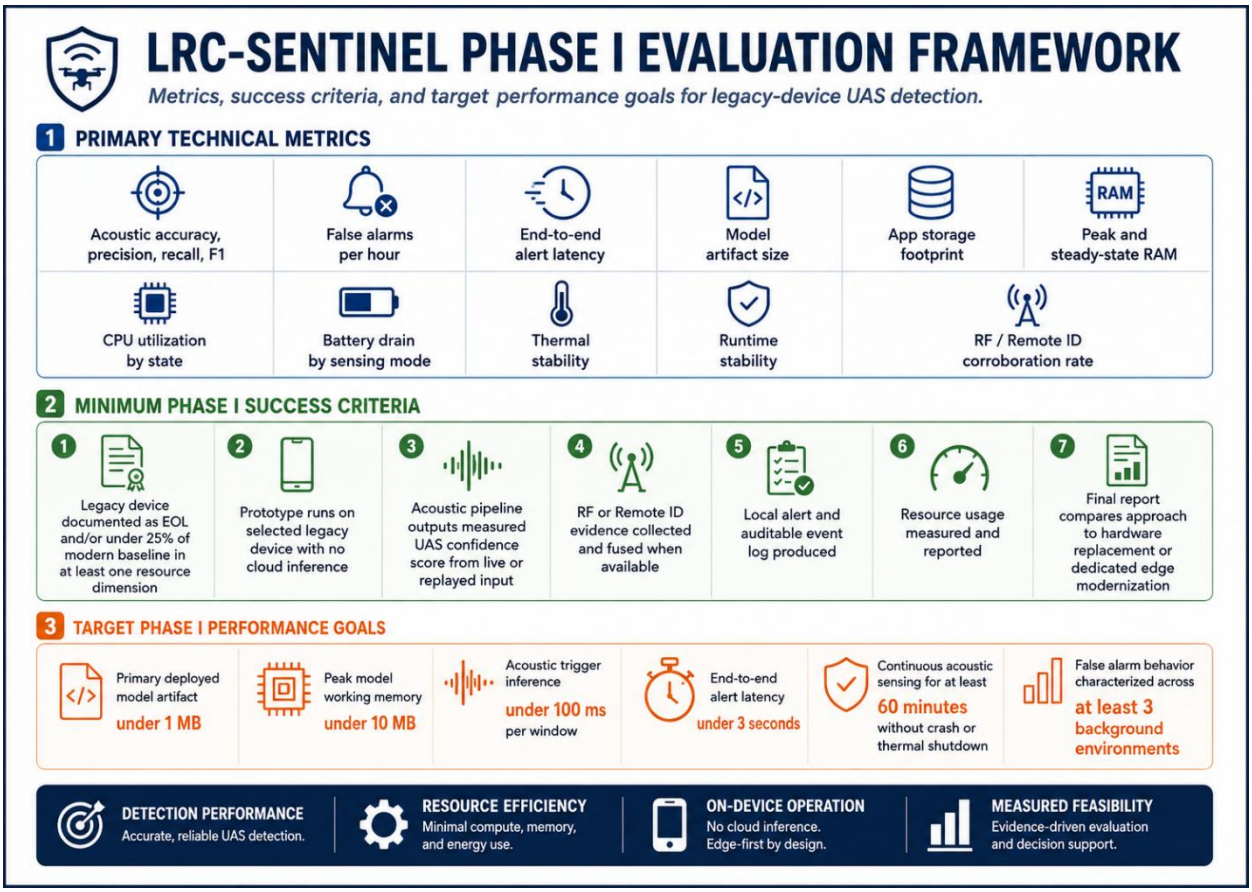
Deliverables:

- Final Phase I report.
- Prototype source and model artifact inventory.
- Evaluation dataset description.
- Benchmark package.
- Phase II expansion and transition plan.
- Comparison to state-of-the-art modernization options.



7. Evaluation Metrics and Success Criteria

Phase I success will be determined by measured feasibility on the selected low-resource device. The evaluation will not rely on offline accuracy alone. It will combine detection performance and resource performance.



These are Phase I feasibility targets, not final operational requirements. The program goal is to prove the upcycling method and establish a credible path to a Phase II operational prototype.

8. Risks, Mitigations, and Phase II Transition

Technical Risks

The first risk is acoustic false alarms. Small drone sounds may overlap with lawn equipment, fans, vehicles, generators, or wind. LRC-Sentinel mitigates this through negative background collection, synthetic augmentation, harmonic and temporal features, temporal smoothing, and RF corroboration.

The second risk is Android RF variability. Older Android devices differ in Bluetooth, Wi-Fi, Remote ID, and scan behavior. LRC-Sentinel mitigates this by treating acoustic sensing as the primary path and RF as corroboration. The system will document exact RF capabilities of the selected device rather than assuming ideal radio access.

The third risk is insufficient compute headroom. A legacy device may not support continuous neural inference. LRC-Sentinel mitigates this through staged execution, classical ML baselines, fixed-size buffers, quantization, pruning, and triggered inference.

The fourth risk is dataset mismatch. Public datasets may not match the selected device microphone or operating environment. LRC-Sentinel mitigates this with local data collection, device-specific calibration, negative background capture, controlled augmentation, and on-device replay testing.

The fifth risk is over-scoping. Counter-UAS is a broad mission area. LRC-Sentinel limits Phase I to passive detection and alerting, not defeat, tracking, interdiction, or air-defense integration.

Phase II Transition

A successful Phase II will prove that computational upcycling is repeatable across multiple low-resource systems, not a one-off optimization for one Android device. Phase II will expand the approach to at least three platforms.

Phase II will mature the semantic overlay into a reusable low-resource sensing framework. It will expand datasets, improve RF front-end support, harden the Android runtime, validate longer-duration sensing, and improve transition interfaces. Phase II will also evaluate integration with tactical awareness systems, enterprise device management, and secure update workflows.

The broader transition value is not limited to UAS detection. The same method can support vehicle-health monitoring, acoustic anomaly detection, RF environment awareness, perimeter sensing, logistics monitoring, and reuse of ruggedized devices that would otherwise be retired. The commercial and government value proposition is upcycling: extracting new mission function from hardware already purchased, tested, and understood.

Groundbreaker Solutions is well positioned to execute this Phase I because the work combines .NET and Android application architecture, Python model development, edge deployment, systems integration, and defense-oriented prototype planning. The team will focus the Phase I on measurable feasibility, credible constraints, and a demonstration that reviewers can understand immediately: an old rugged Android device, running locally, detecting UAS signatures, fusing acoustic and RF evidence, and emitting an actionable alert without being replaced.

9. Team and Organization

Jason L. Lind will serve as Principal Investigator and technical lead for LRC-Sentinel, bringing more than 25 years of software architecture, systems integration, and defense-relevant prototype development experience to the Phase I effort. His background spans .NET, Python, Android/mobile architecture, Azure cloud systems, AI-assisted workflows, signal-processing prototypes, secure application design, and rapid SBIR/RFI concept development. As founder of Groundbreaker Solutions LLC and a U.S. Air Force veteran with avionics backshop experience, Mr. Lind is well suited to lead a computational-upcycling effort that bridges legacy hardware constraints, tactical mission need, and deployable software engineering. For this proposal, he will guide the low-resource device characterization, Python model-development workflow, Android runtime architecture, acoustic/RF evidence fusion, evaluation metrics, and Phase II transition planning across multiple low-resource platforms.

Jack R. Driscoll will support LRC-Sentinel as Hardware and RF Subject Matter Expert, advising the team on RF sensing assumptions, wireless collection constraints, Remote ID considerations, COTS RF front-end options, and practical interpretation of Wi-Fi, Bluetooth, BLE, and RF metadata from legacy Android-class hardware. As Chief Engineer at Groundbreaker Solutions LLC, Mr. Driscoll brings deep experience in RF and wireless-network engineering, including low-latency and high-reliability Wi-Fi, LTE, 5G, BLE, and mesh-network topologies for contested, infrastructure-poor, and DDIL environments. His background in signal propagation, interference mitigation, antenna/link-budget design, network-security protocols, embedded Linux systems, RF-signal identification, edge AI, ROS2-based autonomy stacks, Jetson-class edge compute, and field-deployed DevSecOps workflows will help keep the Phase I RF approach technically credible and transition-relevant. For this effort, Mr. Driscoll will contribute to hardware integration, RF/BLE test planning, feature selection, COTS RF or SDR corroboration options, device-validation activities, and acoustic/RF fusion logic, ensuring that LRC-Sentinel demonstrates useful UAS detection while avoiding overclaims about raw RF/IQ access from built-in mobile radios.

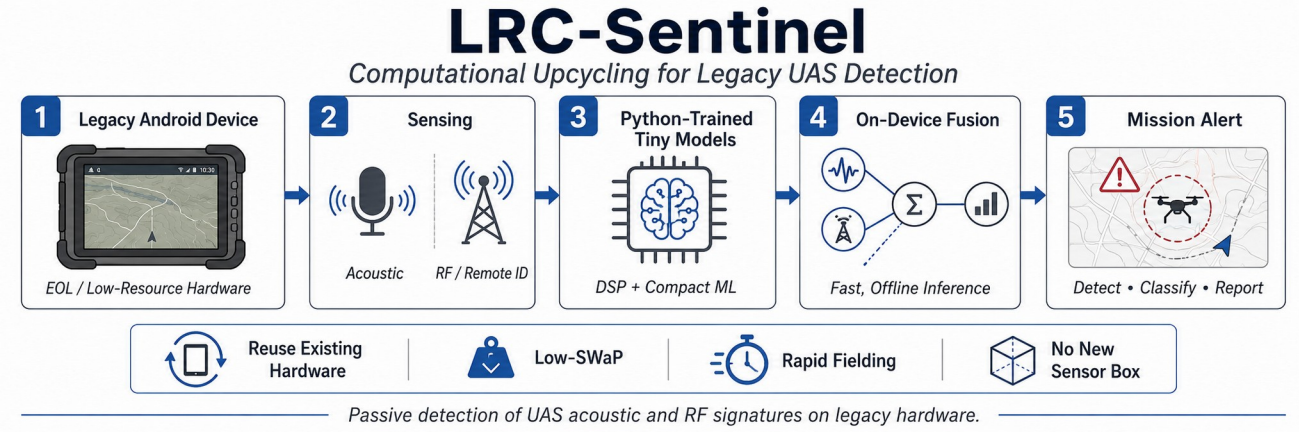
Groundbreaker Solutions LLC is a small U.S.-owned technology company focused on rapid prototype development, AI-enabled software systems, secure edge/cloud architectures, and defense-relevant modernization concepts. The company brings practical experience across .NET, Python, Android/mobile development, Azure cloud services, embedded and edge-device integration, signal-processing prototypes, cybersecurity, and SBIR/RFI response development. For LRC-Sentinel, Groundbreaker Solutions will apply this cross-domain engineering background to demonstrate computational upcycling of legacy low-resource hardware for passive UAS acoustic and RF detection. The team’s approach emphasizes measurable feasibility, reusable software architecture, resource-aware model deployment, secure device operation, and transition planning from a Phase I Android surrogate demonstration to Phase II validation across multiple low-resource platforms.

1. What are we trying to do?

Directly aligned to Low Resource Computing: reuse existing DoW assets instead of replacing hardware.

Build a Phase I prototype that upcycles a legacy rugged Android tablet/handheld into a passive UAS detection node.

Reuse EOL or <25% resource hardware
Detect UAS acoustic signatures using onboard microphone
Corroborate with Android-native RF / Remote ID evidence
Fuse evidence locally and emit actionable alert logs / TAK-compatible events



No new sensor box | **No cloud inference** | **Measured feasibility**

End state: old device + compact models + bounded-memory runtime = useful UAS warning capability years earlier than normal modernization.

2. Technology and commercial product

A deployable low-resource sensing software stack, not a new radio or new compute architecture.

LRC-Sentinel Product Target

Legacy Android Runtime

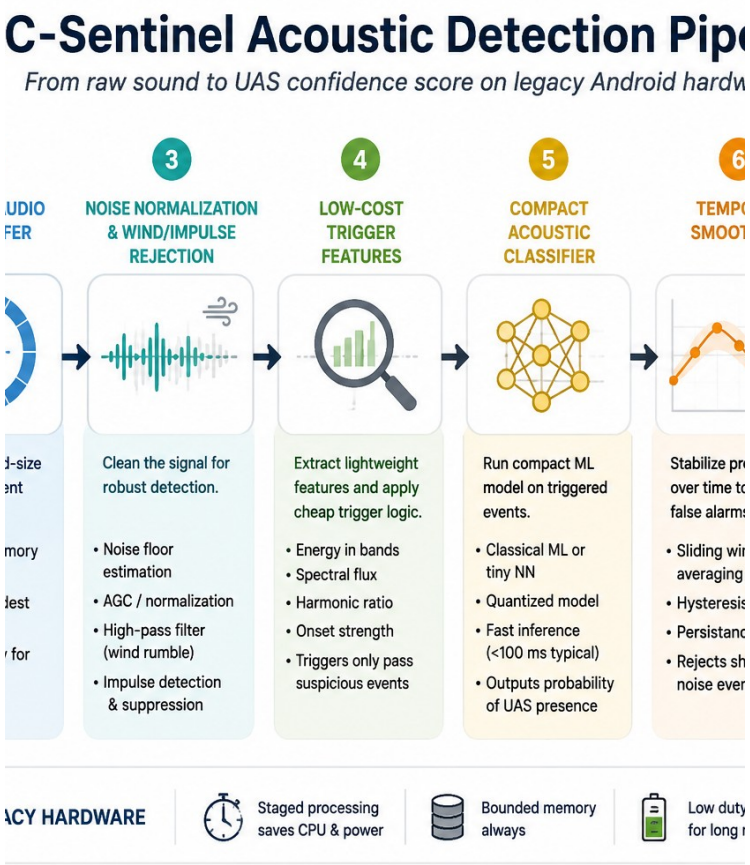
Kotlin/Java + optional NDK runtime for bounded audio buffers, RF/BLE scans, local inference, fusion, logging, and alert output.

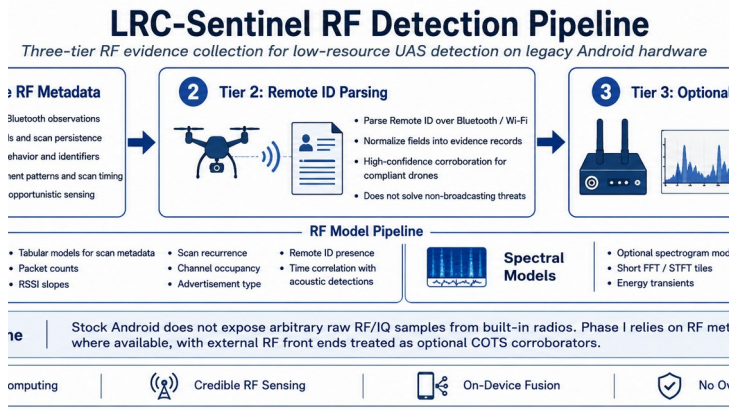
Python ML Toolchain

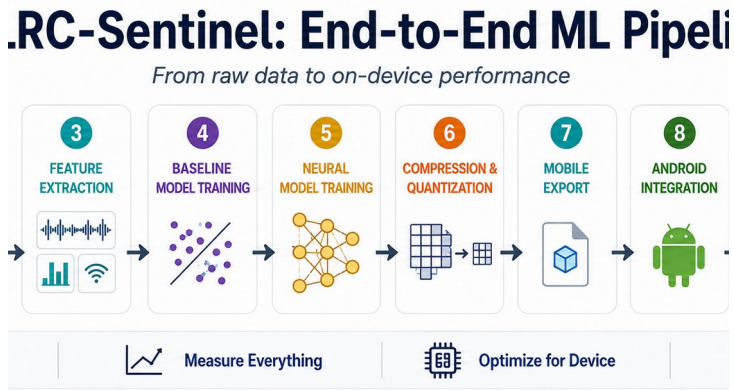
Training, feature extraction, augmentation, model compression, calibration, and export to TensorFlow Lite / ONNX Mobile / compact classical artifacts.

Transition Interfaces

Local dashboard, auditable event log, benchmark package, and optional TAK/ATAK-compatible event output for operator-facing demonstrations.







Phase I produces: prototype app, model artifacts, device characterization, benchmark results, final report, and Phase II expansion plan.

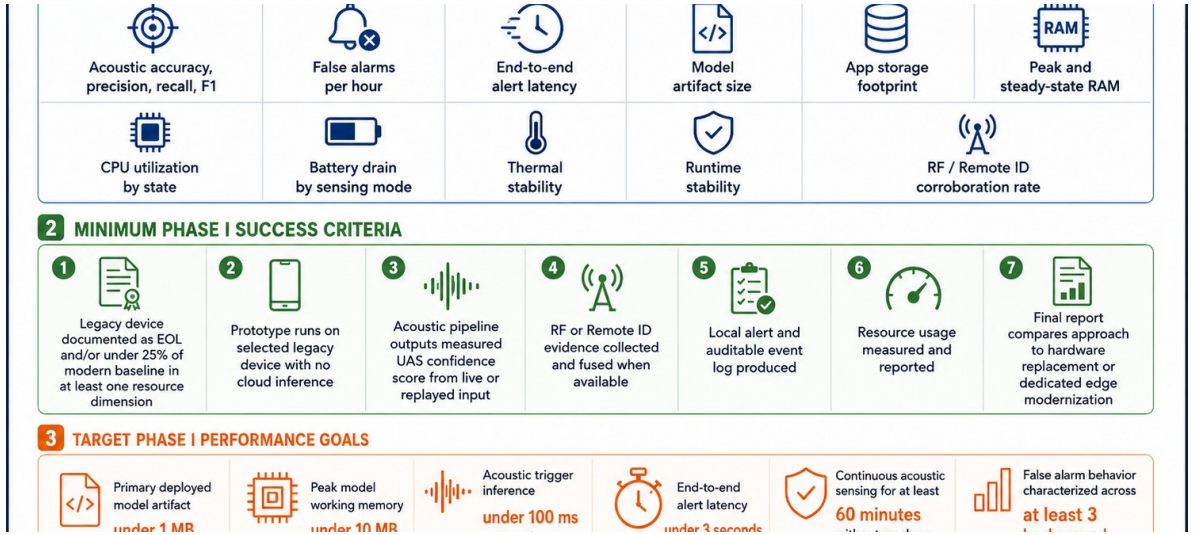
3. Current approach and market limitations

Today, new capability usually means new hardware; LRC-Sentinel tests whether software can reclaim existing assets.

Today: Replace or Add Hardware	Research & Product Landscape	Gap LRC-Sentinel Targets
Dedicated C-UAS sensors, new edge devices, new radios, or platform modernization intervals. Works technically, but often slow, costly, and integration-heavy.	Acoustic UAS detection, RF sensing, Remote ID, edge AI, tinyML, SDR front ends, and mobile tactical awareness systems exist as separate technology lanes.	Few offerings prove net-new sensing on EOL / low-resource fielded devices using measured RAM, CPU, storage, battery, thermal, and OS constraints.

Key limitations in the marketplace

Hardware replacement creates schedule and certification drag
Stock Android lacks raw RF/IQ access; overclaiming is common
Cloud inference is brittle for DDIL and tactical conditions
Generic AI models often exceed legacy resource budgets



Differentiator: measured computational upcycling - compact acoustic + RF evidence fusion that runs locally on constrained legacy devices.

4. Management, facilities, and qualifications

Small, accountable team with software architecture, RF/hardware integration, and rapid prototype execution experience.

Jason L. Lind - PI / Technical Lead

25+ years software architecture and systems integration; USAF avionics backshop experience; leads device characterization, Python ML pipeline, Android runtime, fusion logic, metrics, and Phase II planning.

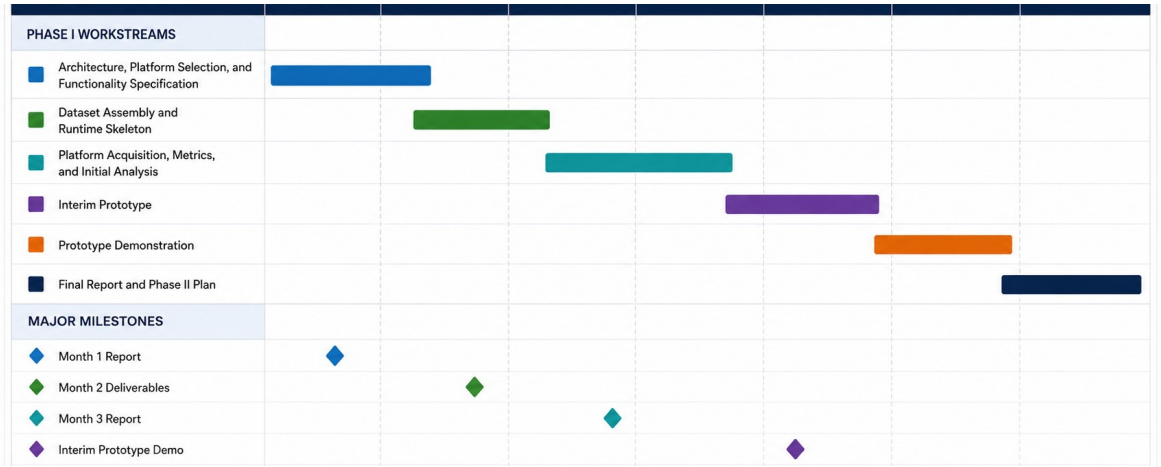
Jack R. Driscoll - Hardware / RF SME

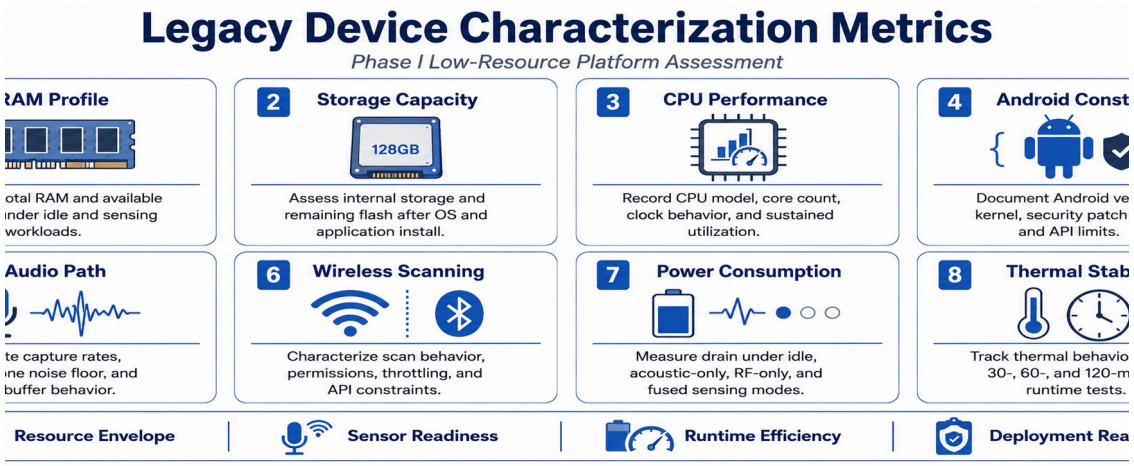
Chief Engineer; RF and wireless-network engineering across Wi-Fi, LTE, 5G, BLE, mesh, DDIL, embedded Linux, antenna/link budgets, COTS RF/SDR corroboration, and device validation.

Groundbreaker Solutions LLC

Small U.S.-owned technology company focused on rapid prototypes, AI-enabled software, secure edge/cloud architectures, Android/mobile, Python, embedded integration, and SBIR/RFI execution.

Execution model: build, measure, compress, deploy, benchmark - every month ends with a concrete artifact.





5. Technical Summary Quad Chart

LRC-Sentinel: computational upcycling for passive UAS acoustic/RF detection on legacy Android hardware.

Technical Objective

Prove that a low-resource legacy Android platform can run passive UAS acoustic detection, RF/Remote ID corroboration, and local fusion without cloud inference or hardware replacement.

Innovation / Approach

Semantic overlay: platform characterization, bounded-memory acoustic pipeline, Android-native RF metadata, optional COTS RF corroboration, compact Python-trained models, and explainable alert fusion.

Phase I Deliverables

Month 1 architecture and metrics plan; Month 3 device characterization and initial models; Month 4 interim prototype; Month 5 on-device demo; Month 6 final report and Phase II plan.

Team / Transition

Groundbreaker Solutions: rapid prototype, Android, Python, edge/cloud, cybersecurity. Jason Lind leads technical execution; Jack Driscoll leads hardware/RF validation. Phase II expands to 3 low-resource systems.

Target Phase I proof: <1 MB detector, <10 MB model working memory, <100 ms trigger inference, <3 sec alert latency, 60+ min continuous sensing.