TECHNICAL WHITE PAPER · AIRWAVE

Airwave: A Full-Spectrum SDR Scanner and Emitter Classifier for iPadOS

A tablet and a $30 RTL-SDR dongle as a complete spectrum-sensing instrument: vDSP windowed FFT, Welch periodogram averaging, robust noise-floor estimation, ITU 99% occupied-bandwidth measurement, a Metal-rendered waterfall, and band-plan-aware emitter classification from 24 MHz to 1.766 GHz.

RTL-SDR | vDSP / Accelerate | Welch PSD | Metal Waterfall | CoreBluetooth | 24 MHz – 1.766 GHz

VERSION 1.0 — AUGUST 2026

ABSTRACT

Airwave is a native iPadOS application that turns a tablet and a commodity RTL-SDR dongle into a full-spectrum radio-frequency scanner, spectrum analyzer, and automatic emitter classifier covering 24 MHz to 1.766 GHz, supplemented by a Bluetooth Low Energy sensor tab that reaches 2.4 GHz emitters the SDR front end physically cannot tune. The application implements a complete DSP chain in Swift over Apple's Accelerate/vDSP framework — windowed FFT, Welch periodogram averaging, noise-floor estimation, and a run-length signal detector with occupied-bandwidth measurement — feeding a Metal-rendered waterfall display and a band-plan-aware classification layer. This paper surveys the science of spectrum sensing and automatic modulation classification, details Airwave's DSP and detection algorithms, analyzes the platform constraints that shape its architecture (iOS's absence of a Wi-Fi scanning API, the GPL and entitlement barriers to direct DriverKit integration, the R820T2 tuning ceiling), and examines the economic and educational case for accessible spectrum analysis.

01 Introduction

The radio spectrum is a shared, finite, and heavily regulated resource, yet it is invisible to ordinary observation. Understanding what is transmitting in a given location — which bands are occupied, at what power, by what kind of emitter — requires instrumentation that has historically been the province of laboratory spectrum analyzers costing $5,000 to $50,000.

The RTL-SDR revolution changed this cost structure. Repurposing the Realtek RTL2832U DVB-T demodulator chip as a general-purpose I/Q sample source [1] produced a $30 device capable of tuning 24 MHz to 1.766 GHz with 2.4 MSPS of instantaneous bandwidth. A vast open-source ecosystem (GNU Radio, SDR#, GQRX, SDR++) grew around it, but that ecosystem is overwhelmingly desktop-bound: Python and C++ applications requiring a laptop, a keyboard, and a mouse.

Airwave brings full-spectrum sensing to a tablet form factor with a touch-native interface. The design goals are: real-time spectrum and waterfall display at interactive frame rates, automatic detection and measurement of emitters without manual cursor placement, band-plan-aware classification that names what is being received rather than showing an anonymous bump, and complete testability of the DSP chain without hardware in the loop.

02 Scientific Background

2.1 I/Q Sampling and the Complex Baseband

A software-defined radio downconverts a slice of RF spectrum to complex baseband, producing a stream of in-phase (I) and quadrature (Q) sample pairs. The complex-valued sequence $z[n] = I[n] + jQ[n]$ represents the analytic signal, preserving both amplitude and phase — and crucially, allowing positive and negative frequencies to be distinguished, so a 2.4 MSPS complex sample rate captures 2.4 MHz of spectrum centered on the tuned frequency (rather than the 1.2 MHz a real-valued sampler would yield under Nyquist) [2].

The RTL2832U delivers 8-bit unsigned I/Q pairs. Airwave converts these to normalized floats, applies a window function, and computes the FFT via vDSP's optimized complex FFT routines.

2.2 Spectral Estimation: Windowing and Welch Averaging

A raw FFT of a finite sample block implicitly assumes periodicity at the block boundary. When the signal is not periodic in the block length — which it never is — the discontinuity produces spectral leakage: energy from a strong narrowband signal smears across all bins, masking weak signals nearby. Window functions taper the block edges to zero, trading main-lobe width for sidelobe suppression [3]. Airwave applies a Hann window, whose –31 dB first sidelobe and 1.5-bin main lobe represent a standard compromise for general spectrum surveillance.

A single windowed FFT of a noisy signal is itself extremely noisy: the periodogram of white Gaussian noise has a variance equal to its mean regardless of FFT length, so longer FFTs improve frequency resolution but not variance. Welch's method [4] reduces variance by averaging the periodograms of multiple overlapping segments:

$$P_{welch}(f) = (1/K) \cdot \Sigma_{k=1..K} |FFT(w[n] \cdot x_k[n])|^2$$

Averaging *K* independent segments reduces the periodogram variance by a factor of *K*, at the cost of temporal resolution. Airwave's spectrum frames apply Welch averaging to produce a stable noise floor against which threshold-based detection is meaningful — without it, the detector would fire continuously on noise excursions.

2.3 Noise Floor Estimation and Detection Theory

Automatic emitter detection reduces to a hypothesis test at each frequency bin: is the observed power consistent with noise alone (H_0), or does it indicate a signal (H_1)? The classical energy detector compares bin power against a threshold set relative to the noise floor [5]. Setting the threshold at *N* dB above the estimated noise floor fixes the false-alarm probability under a Gaussian noise model; the detection probability then depends on the signal-to-noise ratio.

Airwave's detector uses an 8 dB default threshold above the estimated noise floor. Crucially, the noise floor must be estimated robustly: a simple mean over all bins is biased upward by any strong signals present, raising the threshold and masking weaker emitters. Robust estimation (median or percentile-based) is insensitive to the minority of bins containing signal.

2.4 Occupied Bandwidth and the 99% Power Criterion

Once a signal is detected, characterizing it requires measuring its bandwidth. The ITU-standard definition is *occupied bandwidth*: the frequency span containing 99% of the total emitted power, with 0.5% excluded at each edge [6]. This definition is robust to the shape of the emission — it gives sensible answers for a narrow CW carrier, a wideband digital signal with a raised-cosine roll-off, and an FM broadcast signal with substantial sidebands alike.

Airwave measures occupied bandwidth in noise-subtracted *linear* power. Working in linear rather than logarithmic units is essential: the 99%-power integral is a sum of powers, and summing dB values is meaningless. Subtracting the noise floor before integration prevents the noise pedestal under the signal's skirts from inflating the measured width — a systematic bias that would make every signal appear wider than it is.

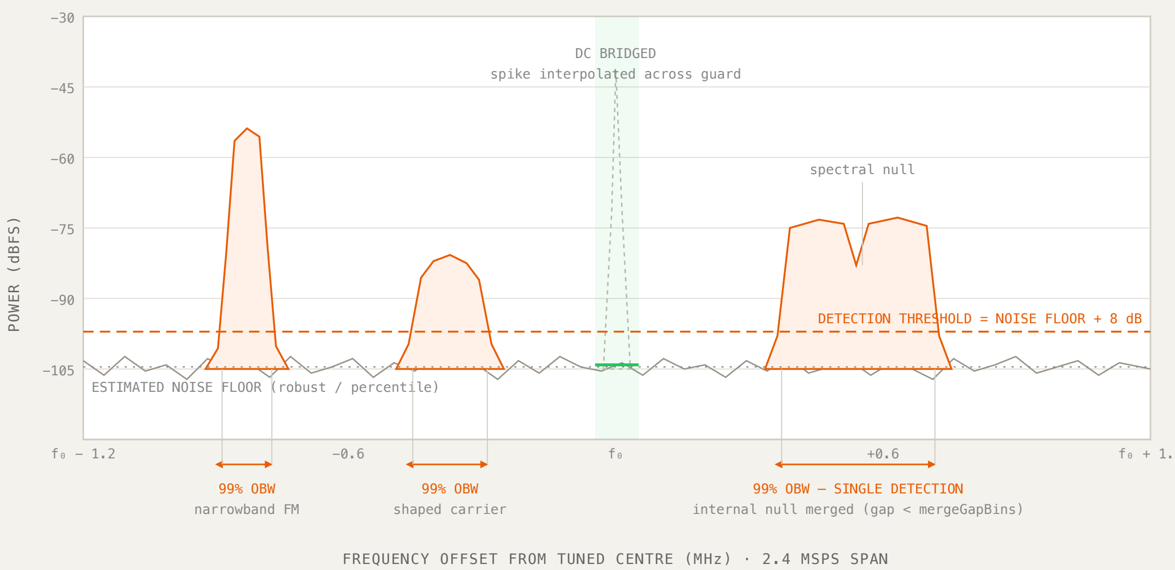

Figure 1. A representative Welch-averaged spectrum frame with the detector's decision structures annotated. Three emitters cross the 8 dB threshold above the robustly estimated noise floor; each is reported with its ITU 99% occupied bandwidth measured in noise-subtracted linear power. At the tuned centre the RTL2832U's permanent DC spike (dashed) has been bridged by linear interpolation across the guard region rather than blanked, so an emitter tuned directly onto centre is not split into two half-width detections. In the wideband emitter at right, an internal spectral null narrower than `mergeGapBins` is merged rather than fragmenting the emission into two reports.

2.5 Signal Detection Engineering: Three Real-World Corrections

The textbook energy detector fails on real hardware in three specific ways that Airwave's implementation addresses explicitly.

DC spike suppression. Every RTL2832U exhibits a permanent DC offset that appears as a large spike at the exact center of the tuned span. Naïvely blanking these bins would split any real signal the user tuned directly onto into two half-width detections. Airwave instead *bridges* the DC region: it replaces the affected center bins with a linear interpolation between the power values just outside the guard region, preserving the continuity of any signal spanning the center while removing the artifact.

Gap merging. Digital carriers routinely contain spectral nulls inside their occupied band — an OFDM signal with unused subcarriers, a shaped carrier with a pilot notch. A run-length detector operating on strict threshold crossings would fragment one emitter into several detections. Airwave merges runs separated by fewer than `mergeGapBins` (default 3) quiet bins, treating them as a single emission.

Span-fraction rejection. If the noise floor estimate drifts (from AGC action or a sudden broadband interferer), the threshold can fall below the actual noise, causing the detector to report the entire span as one enormous "signal." Airwave rejects any detection wider than `maximumSpanFraction` of the visible span. This ceiling must be generous rather than tight: ADS-B at 1090 MHz genuinely occupies roughly two-thirds of a 2.4 MSPS span, so the default is set at 0.9 rather than a more aggressive value that would reject legitimate wideband emissions.

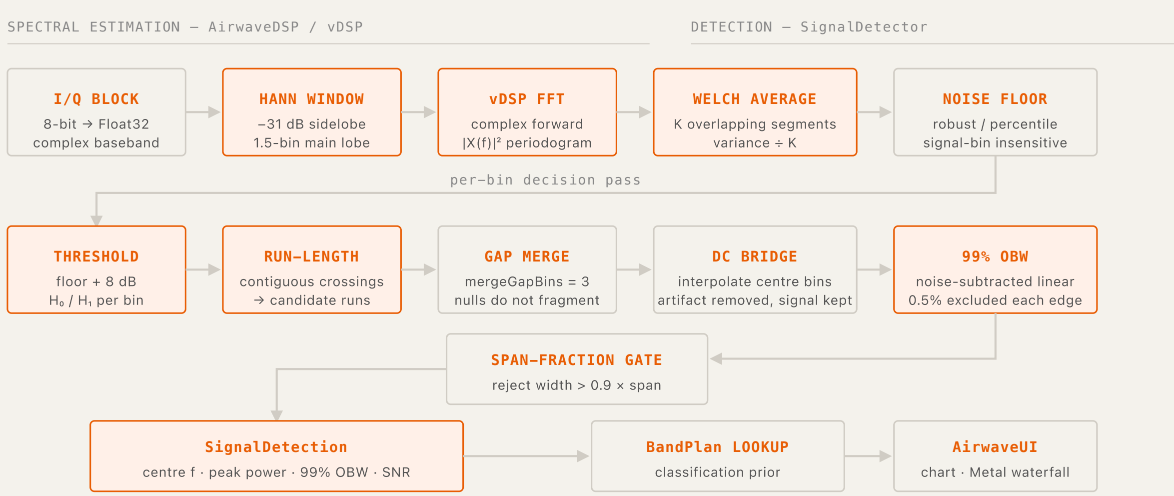

Figure 2. The Airwave DSP and detection chain. Everything from the I/Q block to `SignalDetection` is platform-neutral Swift over Accelerate/vDSP and is unit-tested on macOS with no radio present. The three shaded engineering corrections — gap merging, DC bridging, and the span-fraction gate — are the difference between the textbook energy detector and one that behaves on real RTL2832U hardware.

03 System Architecture

3.1 Module Structure

Airwave is a Swift Package with six library targets plus a development tool. `AirwaveCore` holds the I/Q and tuner models, the US band plan, the signal taxonomy, and the classifier feature vector. `AirwaveTransport` defines the `SDRSource` protocol and provides the rtl_tcp client and server plus a synthetic source. `AirwaveDSP` implements the vDSP FFT, Welch averaging, and the signal detector. `AirwaveBluetooth` wraps CoreBluetooth scanning, advertisement parsing, and classification. `AirwaveUI` supplies the SwiftUI spectrum chart, the Metal waterfall, the BLE radar, and the controls, and `AirwaveApp` is the `@main` entry point. `Tools/rtltcp-sim` is an rtl_tcp server that serves synthetic scenes over real TCP.

Everything below the UI layer is platform-neutral Swift with no UIKit dependency, and is unit-tested on macOS. This is a deliberate structural choice: the DSP and detection logic — the parts where correctness is hardest to eyeball and easiest to get subtly wrong — are exercised by a test suite that runs anywhere, rather than requiring an iPad and a dongle to validate.

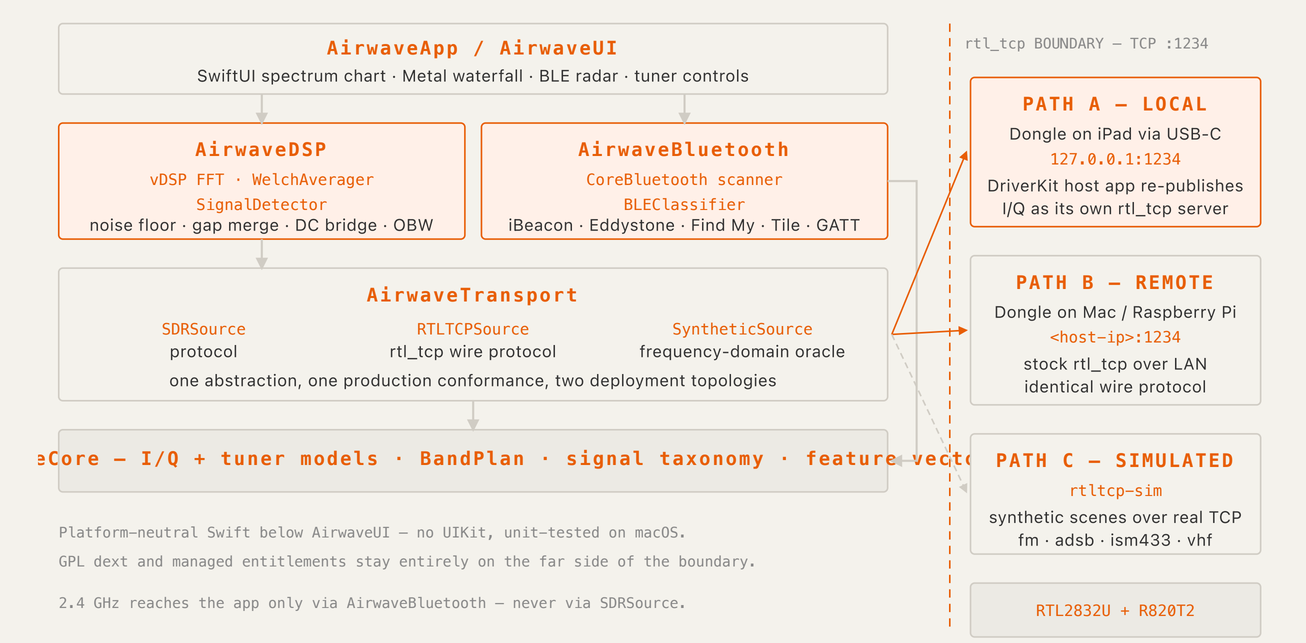

Figure 3. Airwave module architecture. The `rtl_tcp` socket is the hard boundary of the system: everything Airwave ships lives to the left of it, and all three deployment paths — USB-attached dongle on the iPad, remote dongle on a Mac or Pi, and the synthetic simulator — present the identical wire protocol to `RTLTCPSource`.

3.2 Transport: rtl_tcp as the Universal Path

`SDRSource` is the abstraction over I/Q sample sources. Its single production conformance, `RTLTCPSource`, speaks the `rtl_tcp` wire protocol and covers both deployment topologies:

Deployment	Endpoint
Dongle attached to iPad via USB	`127.0.0.1:1234` — the DriverKit host app's own rtl_tcp server
Dongle on a Mac, Raspberry Pi, or homelab node	That machine's IP, port `1234`

DK5AV's USBDriverKit port [7] ships both the DriverKit extension and a host application that re-publishes the I/Q stream as an rtl_tcp server on the iPad itself, so the wire protocol is identical whether the hardware is local or remote.

Why not direct DriverKit integration? Removing the TCP hop would reduce latency marginally, but carries two disqualifying costs. First, the dext is licensed **GPL v2**, which would relicense the entire project and preclude App Store distribution. Second, it requires Apple-approved *managed entitlements*: DriverKit, USB Transport for vendor ID `0x0BDA`, and UserClient Access — each individually approved by Apple, none available on a standard developer account. It additionally requires an M-series iPad; A-series devices cannot run DriverKit extensions at all. The rtl_tcp path has none of these constraints and works identically on every iPad.

3.3 Hardware and Platform Constraints

The 1.766 GHz ceiling. The R820T2 tuner front end has a hard upper limit of 1.766 GHz. This places 2.4 GHz — Wi-Fi, Bluetooth, most ISM device traffic — permanently out of reach of the dongle. Airwave's RF signal taxonomy therefore omits 2.4 GHz emissions entirely rather than listing classifications it can never produce.

No Wi-Fi scanning on iOS. iOS provides no Wi-Fi scanning API at any entitlement level. `NEHotspotHelper` never returned scan results and is deprecated as of iOS 26. The iPad's own Wi-Fi radio therefore cannot serve as a spectrum sensor under any configuration.

CoreBluetooth can. BLE scanning is public API, requires no special entitlement, and yields per-advertisement RSSI plus device identity and service data. This is packet metadata rather than raw spectrum, which is why Airwave presents it as a separate tab with a distinct visualization rather than pretending it is another `SDRSource`.

3.4 The Bluetooth Radar and Honest Uncertainty

The BLE tab plots discovered devices on a radar display with the user at the center and radius derived from RSSI. **Bearing is arbitrary.** A single omnidirectional receiver cannot resolve direction of arrival — that requires an antenna array and phase-difference processing. Rather than plot devices at an arbitrary-but-unlabeled angle that a user would naturally read as directional information, Airwave assigns each device a stable pseudo-angle derived from its identifier (so it does not jitter between scans) and states explicitly in the UI that bearing is not measured.

`BLEClassifier` identifies iBeacon, Eddystone (UID/URL/TLM variants), Find My, Fast Pair, Tile, and GATT service profiles, all parsed from plain advertisement values so the classification rules are unit-tested without a radio present.

Synthetic data labeling. A "Demo data" toolbar option displays synthetic devices for working without hardware. It is always labeled in orange in the UI and is never silently substituted for real scan results — a design commitment that extends to the RF side, where the synthetic source is likewise explicitly indicated.

3.5 The Band Plan as Classification Prior

`BandPlan` encodes the US/ITU Region 2 spectrum allocation, restricted to what the RTL2832U can actually reach. Each `BandAllocation` carries a name, a service category (broadcast, aviation, marine, amateur, public safety, cellular, satellite, industrial, data link, radar), a frequency range, a typical per-emitter channel bandwidth, and an ordered list of expected signal classes.

This structure serves two purposes. As a *display* layer, it labels the spectrum ruler so a detected signal at 162.475 MHz is annotated "NOAA Weather Radio" rather than shown as a bare frequency. As a *classification prior*, the expected-emissions list conditions the classifier: a 16 kHz-wide FM-shaped signal at 146.52 MHz is far more likely to be amateur narrowband FM than anything else, and the band plan encodes that domain knowledge directly.

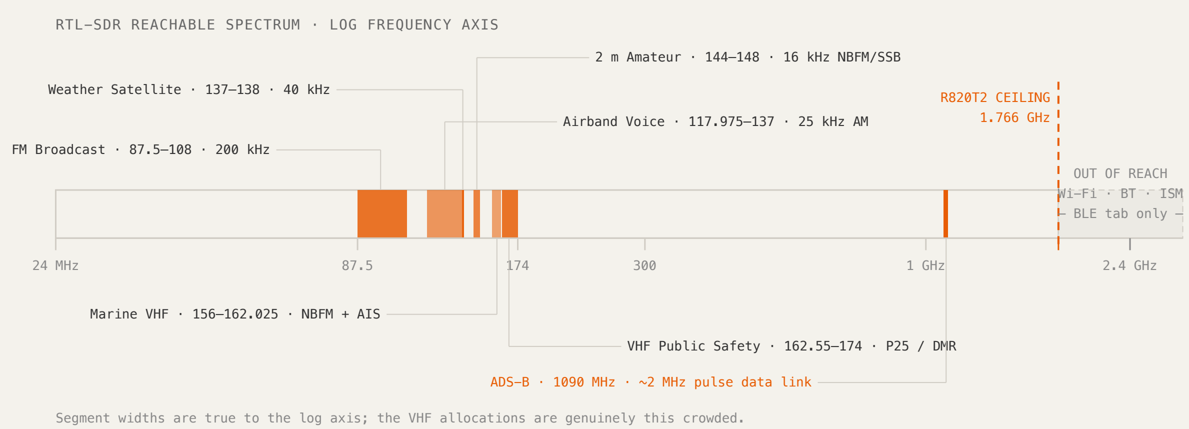

Figure 4. The reachable band plan on a logarithmic frequency axis. Every allocation Airwave classifies falls between 24 MHz and the R820T2's hard 1.766 GHz ceiling; 2.4 GHz is shown greyed at right because no configuration of the dongle can tune it — those emitters reach the application only through the CoreBluetooth tab.

Representative allocation	Range	Channel BW	Expected emissions
FM Broadcast	87.5–108 MHz	200 kHz	FM broadcast
Airband Voice	117.975–137 MHz	25 kHz	AM voice
Weather Satellite	137–138 MHz	40 kHz	Satellite downlink
2 m Amateur	144–148 MHz	16 kHz	NBFM, SSB, digital
Marine VHF	156–162.025 MHz	25 kHz	NBFM, AIS
VHF Public Safety	162.55–174 MHz	12.5 kHz	NBFM, P25, DMR
ADS-B	1090 MHz	~2 MHz	Pulse data link

04 Verification Strategy

4.1 The Synthetic Source as Oracle

Airwave's most consequential testing decision is that `SyntheticSource` constructs each I/Q block **in the frequency domain and inverse-FFTs it**. This means the test knows the exact frequency, bandwidth, and power of every signal it injects — the synthetic scene is a ground-truth oracle, not an approximation.

Tests therefore assert that the detector recovers the exact parameters that were synthesized: a signal placed at 433.92 MHz with 60 kHz occupied bandwidth must be detected at 433.92 MHz ± tolerance with a measured occupied bandwidth of 60 kHz ± tolerance. This closes the loop on the entire DSP chain — window, FFT, Welch averaging, noise-floor estimation, run detection, gap merging, DC bridging, and occupied-bandwidth measurement — against a known answer, rather than eyeballing a waterfall and declaring it plausible.

66

TESTS PASSING — AGAINST A FREQUENCY-DOMAIN ORACLE

Transport, DSP, detection, spectrum and waterfall UI, tuning, BLE scanning and classification, and the radar topology view are working and verified. Because `SyntheticSource` builds each I/Q block in the frequency domain and inverse-FFTs it, the tests assert recovered centre frequency and 99% occupied bandwidth against the exact values that were synthesized. BLE parsing and classification are covered by tests; live BLE scanning has not been exercised on hardware, because the iOS Simulator has no Bluetooth radio.

4.2 Real-Socket Integration Tests

The transport layer's integration tests run the actual `RTLTCPSource` client against a real TCP socket served by `rtltcp-sim`, exercising the genuine wire protocol including command handshakes, sample-rate configuration, and streaming. This catches protocol-level regressions that a mocked transport would miss.

4.3 Hardware-Free Development

The command

```
$ swift run rtltcp-sim ism433
```

serves synthetic signals over the genuine rtl_tcp protocol on port 1234. Any rtl_tcp client — Airwave, SDR++, GQRX — works against it. Available scenes are `fm`, `adsb`, `ism433`, and `vhf`. This makes the entire development loop hardware-independent, and doubles as a cross-validation tool: if SDR++ sees the same signals Airwave does from the same simulated source, the discrepancy is not in Airwave's DSP.

05 Application Domains and Economic Analysis

5.1 Cost Comparison

Instrument	Frequency range	Cost	Portability
Benchtop spectrum analyzer (Keysight / R&S)	9 kHz – 26.5 GHz	$15,000–$80,000	Laboratory only
Handheld analyzer (Anritsu, Signal Hound)	9 kHz – 6 GHz	$3,000–$15,000	Field-portable
RTL-SDR + laptop + SDR#	24 MHz – 1.766 GHz	$30 + laptop	Requires laptop
iPad + RTL-SDR + Airwave	**24 MHz – 1.766 GHz (+ BLE 2.4 GHz)**	**$30 + owned iPad**	**Fully handheld**

1000×

COST DIFFERENTIAL VS. LABORATORY INSTRUMENTATION

For applications inside the RTL-SDR's frequency and dynamic-range envelope, a $30 receiver on an already-owned tablet delivers spectrum measurement that has historically required a $15,000–$80,000 benchtop analyzer — three orders of magnitude, with the instrument fitting in one hand.

5.2 Use Cases

Amateur radio. Band-condition monitoring, antenna performance verification (observing a transmitted signal's actual occupied bandwidth), interference hunting, and satellite pass reception in the 137 MHz weather-satellite band.

RF interference investigation. Locating the source of interference to a receiver — identifying whether a noise floor elevation is broadband (switching power supply, LED lighting) or a specific narrowband emitter, and characterizing its bandwidth and duty cycle.

Regulatory and compliance familiarization. Observing which allocations are actually in use in a given location, and verifying that an emission conforms to its allocation's expected channel bandwidth.

STEM education. The spectrum is an abstraction until it is visible. A tablet-based waterfall showing FM broadcast stations as discrete 200 kHz blocks, aircraft transponders pulsing at 1090 MHz, and a garage door opener at 433 MHz makes the invisible concrete in a way no textbook diagram achieves. The synthetic scene generator allows classroom use with no hardware.

Situational awareness. Understanding the local RF environment before deploying a wireless system — identifying occupied channels, characterizing the noise floor, and finding clear spectrum.

5.3 Legal and Ethical Framing

Airwave is a receive-only instrument; it cannot transmit. In the United States, reception of most radio transmissions is legal, with statutory exceptions notably including cellular telephone communications (Electronic Communications Privacy Act, 18 U.S.C. § 2511) and encrypted transmissions. Airwave's band plan does not include decoders for restricted services, and the application performs spectral analysis — measuring where and how wide emissions are — rather than demodulating content. Users remain responsible for compliance with the regulations of their jurisdiction, which vary substantially internationally.

06 Limitations and Roadmap

Frequency coverage. Bounded by hardware at 1.766 GHz. Extending above this requires different hardware (HackRF at 6 GHz, Airspy, USRP) with correspondingly different transport integration — the `SDRSource` protocol is the abstraction point where that would occur.

Dynamic range. The RTL2832U's 8-bit ADC provides approximately 48 dB of theoretical dynamic range, substantially less than the 90+ dB of laboratory instrumentation. Strong signals can desensitize the receiver and generate intermodulation products that appear as spurious emissions. This is a hardware limit, not a software one.

Not yet built. Core ML-based automatic modulation classification — the feature-vector infrastructure exists in `AirwaveCore` but is not yet feeding a trained model — along with CoreLocation + MapKit geographic signal-strength mapping, a SwiftData session store for recording and replaying scans, and GeoJSON/CSV export.

Build tooling. `xcodebuild` fails on the development machine with an `Exec format error` during `ComputePackagePrebuildTargetDependencyGraph` — a toolchain-level failure that reproduces on projects with no SwiftPM packages at all. `swift build`, `swift test`, and `swiftc` all work, which is why the project ships `scripts/build-ios.sh` and deliberately uses native framework targets rather than the SwiftPM package for the Xcode build path.

07 Conclusion

Airwave demonstrates that a complete spectrum-sensing instrument — real-time FFT and waterfall display, robust automatic emitter detection with ITU-standard bandwidth measurement, and band-plan-informed classification — runs comfortably on a tablet with a $30 receiver, delivering capability that until recently required laboratory instrumentation.

Two design commitments distinguish the implementation: the frequency-domain synthetic source that makes the entire DSP chain testable against ground truth without hardware, and the consistent refusal to present information the sensor cannot actually provide — no fabricated bearings on the BLE radar, no 2.4 GHz classifications the tuner cannot reach, no unlabeled synthetic data. As software-defined radio hardware continues to fall in price and rise in capability, the constraint on accessible spectrum analysis is increasingly the interface rather than the receiver, and a touch-native tablet application is a better fit for field work than any desktop SDR application.

// References

[1] Osmocom, "rtl-sdr: Turns your Realtek RTL2832 based DVB dongle into a SDR receiver," 2012. https://osmocom.org/projects/rtl-sdr/wiki

[2] Lyons, R.G., *Understanding Digital Signal Processing*, 3rd ed., Prentice Hall, 2010, ch. 8 (Quadrature Signals).

[3] Harris, F.J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," Proceedings of the IEEE, vol. 66, no. 1, pp. 51–83, January 1978.

[4] Welch, P.D., "The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms," IEEE Transactions on Audio and Electroacoustics, vol. 15, no. 2, pp. 70–73, 1967.

[5] Urkowitz, H., "Energy Detection of Unknown Deterministic Signals," Proceedings of the IEEE, vol. 55, no. 4, pp. 523–531, April 1967.

[6] International Telecommunication Union, "Radio Regulations, Article 1.153: Occupied Bandwidth," ITU-R, 2020.

[7] Viehweger, A. (DK5AV), "RTL-SDR USB DriverKit port for iPadOS," GitHub, 2023. https://github.com/arvedviehweger/RTL-SDR-USB-iPadOS

[8] Apple Inc., "Accelerate — vDSP Fourier Transforms," Apple Developer Documentation, 2024. https://developer.apple.com/documentation/accelerate/vdsp

[9] Apple Inc., "Core Bluetooth," Apple Developer Documentation, 2024. https://developer.apple.com/documentation/corebluetooth

[10] Dobre, O.A., et al., "Survey of Automatic Modulation Classification Techniques: Classical Approaches and New Trends," IET Communications, vol. 1, no. 2, pp. 137–156, 2007.

[11] Federal Communications Commission, "FCC Online Table of Frequency Allocations," 47 C.F.R. § 2.106, revised 2024.

[12] Yucek, T., and Arslan, H., "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications," IEEE Communications Surveys & Tutorials, vol. 11, no. 1, pp. 116–130, 2009.