OULUN AMMATTIKORKEAKOULU

PASSIVE RF SIGNAL DETECTION AND LOCALIZATION USING UNMANNED PLATFORMS

Samuli Pylkkönen
Bachelor's Thesis
Autumn 2025
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology
Option of Embedded Software Development

Author: Samuli Pylkkönen
Title of thesis: Passive RF Signal Detection and Localization Using Unmanned Platforms
Term and year of completion: Spring 2026
Pages: 49 + 4 appendices

Passive radio-frequency (RF) localization supports search operations when visual line-of-sight is limited by terrain, vegetation, or weather. This thesis developed and evaluated a passive RF sensing prototype intended for unmanned deployment, combining received-signal-strength (RSSI) mapping with an optional two-element phase-interferometric Angle of Arrival (AoA) capability. RSSI-to-range theory was derived from log-distance path-loss modelling with shadowing and used to motivate later trilateration and geometry (DOP-like) considerations. (Bensky 2008, 140-146; Haslett 2008, 176-178; Zekavat & Buehrer 2019, 666-667.)

A coherent dual-channel software-defined radio receiver with GNSS tagging was implemented and integrated with Python-based logging and post-processing to produce georeferenced CSV datasets, heatmaps, and AoA-bearing overlays. AoA estimation was based on inter-channel phase difference mapped through a two-element interferometer model with quality gating by coherence and SNR. (Bensky 2016, 224-229; Molisch 2011, 28-29.)

Field evaluation was conducted as an outdoor continuous-wave drive-by demonstration using a fixed transmitter at a known location. A short stationary calibration capture was logged to estimate a constant inter-channel phase bias, and the resulting phase offset was applied during the drive-by run to reduce systematic bearing rotation. (Molisch 2011, 161-162; Bensky 2016, 224-227.) The resulting heatmap concentrated strongest received levels near the closest-pass region, and AoA rays from high-coherence samples oriented predominantly toward the reference transmitter area, indicating qualitatively interpretable localization cues. The main limitations were the ground-vehicle platform, the controlled CW reference signal, and hand-held antenna baseline/orientation variation, which increase AoA spread and reduce repeatability. Development priorities include a rigid baseline fixture, repeated calibration checks, improved GNSS robustness, and multi-pass surveys with geometry assessment.

CONTENTS

GLOSSARY

In this thesis following acronyms and technical terms are used.

concept	definition
ADC	Analog-to-Digital Converter; circuit that converts a continuous-time, continuous amplitude signal into discrete digital samples.
AoA	Angle of Arrival; estimated direction from which a radio signal arrives at a receiving antenna array, usually obtained from phase or amplitude differences between elements.
CSV	Comma-Separated Values; plain-text table format where each row is a line and columns are separated by commas.
CW	Continuous wave
dB	Decibel; logarithmic unit that expresses a ratio of powers of amplitudes, commonly used for signal levels and gains
dBi	Decibel relative to an isotropic radiator; antenna gain referenced to an ideal isotropic antenna.
dBFS	Decibels relative to Full Scale; digital power unit that expresses signal level relative to the maximum value representable by an ADC.
DOP	Dilution of Precision; measure describing how receiver-satellite or receiver-anchor geometry amplifies measurement errors into position errors.
DSP	Digital Signal Processing; numerical processing of sampled signals, for example filtering, FFT computation and power estimation.
FFT	Fast Fourier Transform; efficient algorithm to compute the discrete Fourier transform, used here to estimate power in selected frequency bands.

FPGA	Field-Programmable Gate Array; reconfigurable digital logic device used in the bladeRF platform for high-speed signal processing.
GDOP	Geometric Dilution of Precision; DOP metric that combines position and time components into a single geometry indicator.
GNSS	Global Navigation Satellite System; generic term for satellite positioning systems such as GPS, Galileo, BeiDou and GLONASS.
GPIO	General-Purpose Input/Output; digital pins on the Raspberry Pi used for low-speed control and timing signals such as 1 PPS.
GPS	Global Positioning System.
HAT	Hardware Attached on Top; Add-on board format for the Raspberry Pi that connects via the 40-pin GPIO header.
HTML	HyperText Markup Language.
IBW	Instantaneous Bandwidth; contiguous frequency span that the receiver can process at one time.
I/Q	In-phase and Quadrature; pair of orthogonal components representing a complex baseband signal used in SDR receivers.
L1	Primary GNSS frequency band around 1575.42 MHz used for civil navigation signals.
L5	GNSS frequency band around 1176.45 MHz providing higher-accuracy and safety-of-life signals.
LNA	Low-Noise amplifier; first active stage in the receive chain designed to provide gain with minimal added noise.
MS/s	Mega Samples per Second; sampling rate unit equal to one million samples per second.
PD	Power Delivery; USB power-delivery negotiates higher voltages and currents between a power source and a device.

PPS	Pulse Per Second; timing signal that marks exact second boundaries and is used to synchronize timestamps to GNSS time.
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTK	Real-Time Kinematic; high-accuracy GNSS technique that uses carrier-phase and correction data to achieve centimeter-level positioning.
SDR	Software Defined Radio.
SMA	SubMiniature version A; threaded RF connector type commonly used for antennas and coaxial cables.
SSD	Solid-State Drive.
UAV	Unmanned Aerial Vehicle;
USB	Universal Serial Bus.
VSWR	Voltage Standing Wave Ratio; measure of impedance matching between an antenna and feed line, related to reflection losses.

1 INTRODUCTION

Search operations for missing persons in forested or otherwise obstructed terrain are constrained by visibility, access and weather. Optical and thermal sensors depend on a reasonably clear line-of-sight, and their performance is degraded by snowfall, fog, darkness and complex terrain. In Arctic and sub-Arctic conditions, cold temperatures and rapidly changing weather further limit how long search teams can safely operate on the ground or in the air.

Recent incidents in Finland illustrate the consequences when rescue teams cannot quickly locate people in distress. In January 2024, an avalanche in the Pallastunturi fells in Finnish Lapland led to the deaths of a mother and child despite extensive multi-agency search efforts that were hindered by deep snow, very low temperatures, high winds and darkness (Yle News 2024). Such cases highlight how quickly survivability can deteriorate in harsh weather and how critical rapid localization is when conventional visual methods are limited.

At the same time, Finnish authorities are adopting drones as “first-on-scene” tools. The police already use unmanned aircraft extensively for searches and situational assessment, and will evaluate a remotely operated drone-in-a-box system designed to take off in minutes of an alarm and provide live imagery of an incident area before patrols arrive (Keski-Korpela 2025). These developments indicate that unmanned platforms are becoming a standard part of Finnish search and rescue and offer natural carrier for additional sensors such as a passive RF receiver.

This thesis investigates a practical, lower-cost and faster method for locating missing persons by exploiting the RF emissions of mobile phones carried by individuals such as hikers. The approach is fully passive: rather than transmitting any signal, the system measures the Received Signal Strength Indicator (RSSI) from existing cellular or other handset transmissions and uses these measurements to infer the likely location of the device. Outdoor RSSI samples from multiple positions are interpreted through the log-distance path-loss model with log-normal shadowing, enabling approximate range estimation, which is then

used in trilateration to obtain a position estimate (Bensky 2008, 140-146; Haslett 2008, 176-178). Localization geometry and Dilution of Precision (DOP) are used to characterise how measurement noise and anchor geometry translate into coordinate uncertainty (Bensky 2008, 180-184; Zekavat & Buehrer 2019, 666-667).

At high level, the prototype consists of a drone-mounted RF receiver and a GNSS-equipped processing unit that record time-tagged RSSI and, when available, Angle-of-Arrival information from handset-type signals. After a scan of the search area, these georeferenced measurements are processed into RF power heat maps and triangulation-based position estimates that indicate the most probable location of the emitter (Bensky 2008, 140-146; Bensky 2016 224-229).

The main objective of this thesis is to design and implement a drone-based passive RF search prototype and to evaluate its suitability for locating a handset-like RF source in outdoor environments. The focus is on integrating commercially available SDR hardware, lightweight computing and GNSS into a coherent system that can support search operations and produce interpretable localization outputs for field personnel.

To support this objective, the thesis addresses the following research questions:

- How accurately can a passive, drone-mounted RF sensing system localize a handset-type emitter using RSSI-based ranging and trilateration in realistic outdoor conditions (Bensky 2008, 140-146; Haslett 2008, 176-178)?
- How does the geometry of measurement points, as captured by DOP-like metrics, influence the localization uncertainty of the estimated position (Bensky 2008, 180-184; Zekavat & Buehrer 2019, 666-667)?
- To what extent is it technically feasible to obtain reliable AoA measurements from a small two-element phase interferometer mounted on a multirotor unmanned aircraft, and what accuracy and stability can be achieved under realistic flight conditions compared

with static installations (Bensky 2016, 224-229; Balanis 2016, 703-706)?

The scope of this work is limited to outdoor localization of a single passive RF source that behaves similarly to a mobile handset, using a single unmanned aircraft as a mobile sensor platform. The prototype relies on RSSI-based ranging and optional AoA estimates built on standard log-distance path-loss and log-normal shadowing models and is treated as effectively two-dimensional; complex urban propagation, regulatory analysis and multi-user interference management are excluded from the scope (Bensky 2008, 144-146; Haslett 2008, 176-178).

The project is conducted in cooperation with Millog Oy, which provides technical guidance and access to testing equipment. Legal and regulatory aspects and multi-user interference management are outside the scope of this thesis.

The thesis is structured as follows. Chapter 2 introduces the propagation and localization background and underpins the prototype. Chapter 3 describes the hardware platform, including the software-defined radio, antennas, low-noise amplifiers, host computer and GNSS module. Chapter 4 presents the implementation on the unmanned platform and the data-processing pipeline from RF measurements to maps and position estimates. Chapter 5 presents experimental results on localization accuracy and robustness, and Chapter 6 discusses the findings, limitations and directions for further development of drone-based passive RF search systems.

2 SYSTEM OVERVIEW

This chapter explains the design of a reliable drone-based RF search system; first, it is important to understand how outdoor RSSI measurements translate into ranges and, ultimately, positions. Section 2.1 therefore develops the core propagation and localization equations drawing on the log-distance path-loss model, statistical shadowing, and trilateration geometry which then directly inform the selection of antennas, SDR sampling rates, and GPS accuracy.

Building on this theoretical background, Chapter 3 presents the actual hardware platform: the Nuand bladeRF SDR, Raspberry Pi 5 host, low-noise amplifiers, receive antennas and dual-band GNSS module, and explains how their specifications satisfy the requirements derived here.

The main technical components and hardware setup used in the passive RF signal detection and localization project are presented. The reason for selecting each component is explained, along with a description of their key specifications.

2.1 Overview of passive RF signal detection

Passive detection in this thesis means measuring the received power of whatever RF transmissions are present, without any cooperation from the source. Such signals can be detectable even without line-of-sight, because diffraction around terrain and foliage allows energy to bend past obstacles (Haslett 2008, 50-61).

By sampling (RSSI) values from multiple known inspection points and fitting the log-distance path-loss model with log-normal shadowing term, power readings can be converted into approximate ranges (Bensky 2008, 144-146). Position is then estimated by trilateration, in other words intersecting the range circles around anchors with known coordinates (Bensky 2008, 29-31). Section 2.1.1 introduces the free-space Friis baseline and the log-distance path-loss law; section 2.1.2 adds statistical shadowing and shows how to invert RSSI to range; section 2.1.3 outlines the trilateration geometry; section 2.1.4 quantifies

uncertainty via error propagation and Dilution of Precision (DOP) (Bensky 2008, 183-185; Zekavat & Buehrer 2019, 666-667); and section 2.1.5 presents an Angle-of-Arrival (AoA) option using small phase interferometer, which can be fused with RSSI-based ranges (Bensky 2016, 224-226, 228-229).

2.1.1 Log-distance path-loss model

This subsection introduces the free-space baseline: how received power scales with distance and antenna gains, which is used as the starting point for later models.

The Friis relation expresses received power in free-space as a function of transmit power, antenna gains, wavelength and distance. Received power in free-space follows the Friis transmission law in formula 1

$$P_r(\mathrm{d}) = P_t * G_t * G_r \left(\frac{\lambda}{4\pi d}\right)^2 \qquad \text{FORMULA 1}$$

This is the standard free-space law and applies in the antenna far field (as in $d \gg \lambda$ and beyond the Rayleigh/Fraunhofer distance), assuming line-of-sight and matched polarization (Molisch 2011, 48-49).

Symbols for formula 1
P_r = received power (W)

P_t = transmit power (W)

G_t = transmit-antenna gain (linear)

G_r = receive-antenna gain (linear)

d = distance between antennas (m)

λ = wavelength (m).

The decibel form isolates the distance dependence into $-20\log_{10}(d)$ term, which is convenient for engineering calculations. Usually engineers work with decibels:

taking $10log_{10}$ of both sides of (Formula 1) and expressing powers in dBm and antenna gains in dBi yields the dB form; the squared propagation factor produces the $-20\log_{10}(4\pi d/\lambda)$ term.

$$P_r(d)\,[dBm] = P_t[dBm] + G_t[dBi] + G_r[dBi] - 20\log 10\,(\frac{4\pi d}{\lambda}) \quad \text{FORMULA 2}$$

which means every time the distance d increases tenfold, received power drops by about 20 dB (Bensky 2008, 140-141; Molisch 2011, 49).

Section 2.1.2 extends this free-space baseline to real environments using the log-distance path-loss model with shadowing term.

2.1.2 From power samples to range estimates

Real terrain departs from free space; the log-distance path-loss law adds an average decay with distance and a random shadowing term X_σ (in dB) to capture environment variability (Bensky 2008, 144-146; Haslett 2008, 176-178). The model is given in formula 3.

$$P_r(d)\,[dBm] = P_r(d_0)\,[dBm] - 10n\log 10(\frac{d}{d_0}) + X_\sigma \quad \text{FORMULA 3a}$$

Ignoring the random term X_σ, rearranging (Formula 3a) gives the range estimate

$$\hat{d} = d_0 * 10^{\frac{P_r(d_0)-P_r(d)}{10n}} \quad \text{FORMULA 3b}$$

This is the standard inversion of the log-distance model used in RSS-based ranging. In practice, $P_r(d_0)$ and *n* are obtained by calibration. Typical parameter ranges are $n \approx 2$ outdoors and $n \approx 3-4$ indoor; with log-normal shadowing spreads $\sigma \approx 5-8$ dB so $\hat{d}$ is a median (not per-sample) range (Bensky 2008, 144-146; Haslett 2008, 176-178).

Symbols for formula 3

d_0 = reference distance (often 1 m; Friis/free-space used to define $L(d_0)$)

n = path-loss exponent (≈ 2 outdoors LOS; rising to 3-4 in suburban/indoor)

X_σ = zero-mean log-normal shadowing term with standard deviation $\sigma(\mathrm{dB})$

2.1.3 Localization geometry and trilateration

After converting power samples to approximate ranges, position is obtained by trilateration: the intersection of range circles around anchor locations. Once the RSSI derived ranges r_i are available 2.1.2, the handset's 2-D position (*x, y*) is recovered by solving the circle equations centered on at least three anchors with known coordinates (x_i, y_i). Two circles can intersect in up to two symmetric points; a third range removes this ambiguity and stabilizes the solution (Bensky 2008, 29-31).

Linearization (three-anchor case)

Each range measurement defines a circle centred at an anchor; the target lies at the intersection of these circles.

The circle-based trilateration geometry is introduced in formula 4.

Figure 1 visualises the geometric interpretation of formula 4: each RSSI-derived range defines a circle around an anchor, and the target lies at the intersection region of multiple circles. With realistic range errors, the circle do not intersect at a single point, motivating the linearised least-squares formulation in Formulas 5-6. (Bensky 2008. 29-31, 162-165; Zekavat & Buehrer 2019, 396-397.)

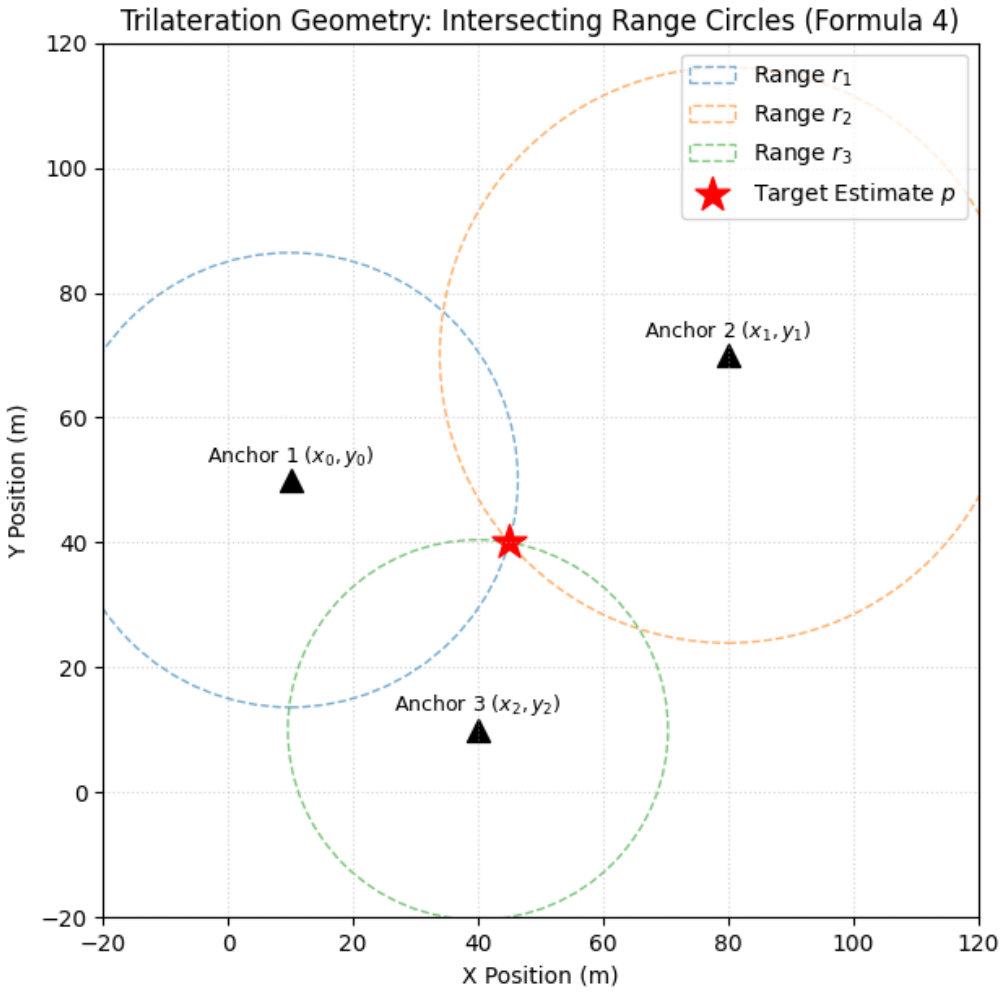

FIGURE 1. Trilateration geometry as intersecting range circles from three anchors (conceptual).

In practice, measurement noise typically prevents exact intersection, so the position estimate is obtained by least-squares multilateration rather than an exact circle intersection.

$$(x - x_i)^2 + (y - y_i)^2 = r_i^2, i = 1, 2, 3 \qquad \text{FORMULA 4}$$

Subtracting the first circle from the others to (a standard linearization step in range-based positioning) cancels the quadratic terms and yields a linear system in (*x, y*) as shown in formula 5 (Bensky 2008, 162-165; Zekavat & Buehrer 2019, 396-397).

$$Ap = b,$$

$$A = \begin{bmatrix} x_2 - & x_1\, y_2 - & y_1 \\ x_3 - & x_1\, y_3 - & y_1 \end{bmatrix}, P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$b = \frac{1}{2}\begin{bmatrix} r_2^1 - & r_2^2 + x_2^2 - & x_1^2 + & y_2^2 - y_1^2 \\ r_1^2 - & r_3^2 + x_3^2 - & x_1^2 + & y_3^2 - y_1^2 \end{bmatrix} \qquad \text{FORMULA 5}$$

Provided the anchors are non-collinear ($\det(A) \neq 0$), the closed-form solution is formula 6a (Bensky 2008, 162-165).

$$p = A^{-1}b \qquad \text{FORMULA 6a}$$

With *N > 3* ranges, stack the linearized equations and solve by weighted least squares to mitigate measurement noise:

$$p = (A^T W A)^{-1} A^T W b \qquad \text{FORMULA 6b}$$

with $W = diag(\sigma_{r_1}^2, \ldots, 1/\sigma_{r_N}^2)$. When all ranges have equal variance, set *W = I* and (Formula 6b) reduces to ordinary least squares (Bensky 2008, 180-184; Zekavat & Buehrer 2019, 666-667).

Symbols for formulas 4-6

x = unknown target *x*-coordinate (m)

y = unknown target *y*-coordinate (m)

x_i = *x*-coordinate of anchor *i* (m)

y_i = *y*-coordinate of anchor *i* (m)

r_i = range (distance) from anchor *i* to target, estimated from RSSI (m)

i = anchor index (*i = 1, 2, 3)*

A = geometry matrix (dimensionless)

p = position vector (m)

b = right-hand side vector (m^2)

$\det(A)$ = determinant of *A* (non-zero when anchors are non-collinear)

A^{-1} = inverse of *A*

W = weighting matrix diag(1/$\sigma_{r_i}^2$) (dimensionless)

N = number of ranges (measurements)

2.1.4 Error propagation and GDOP

Beyond a point estimate, field use needs both an uncertainty and a geometry score; error propagation links range noise to coordinate variance, and DOP summarizes how anchor geometry amplifies errors.

The trilateration in section 2.1.3 returns a point (*x, y*), but responders also need the uncertainty of that point and a geometry score to judge whether to trust it. Error propagation links range noise to coordinate variance, and Dilution of Precision (DOP) summarizes how anchor geometry amplifies errors. (Bensky 2008, 180-184.)

From range noise to coordinate spreads

In the linearized least-squares solution, let *A* be the geometry matrix from (Formula 5); and let σ_R be the standard deviation of a single range measurement. The covariance of the (*x, y*) estimate is formula 7.

$$Cpp = \sigma_R^2 (A^{\mathrm{T}} A)^{-1} \qquad \text{FORMULA 7}$$

Diagonal entries C_{pp} give the coordinate variances σ_x^2 and σ_y^2. (Bensky 2016, 182-184.)

DOP is a dimensionless geometry metric: it is formed from the geometry matrix

$$G = (A^T A)^{-1}, \qquad \text{FORMULA 8a}$$

which depends only on anchor geometry; measurement noise scales the covariance via σ_R^2 but does not enter DOP itself. Using the diagonal elements $g_{xx}g_{yy}g_{zz}g_{tt}$ of *G*,

$$GDOP = \sqrt{g_{xx}+g_{yy}+g_{zz}+g_{tt}} \quad \text{FORMULA 8b}$$

For 2-D RSSI localization (no clock term) the horizontal DOP is

$$HDOP = \sqrt{g_{xx}+g_{yy}} \quad \text{FORMULA 9}$$

This definition matches GNSS/positioning texts, where DOP comes from the inverse normal matrix and reflects only geometry; it is not divided by a noise term. In practice, coordinate standard deviations follow from $\sigma_x = \sqrt{g_{xx}}\sigma_R$ and $\sigma_y = \sqrt{g_{yy}}\sigma_R$ (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667). Spreading anchors around the expected target area reduces DOP, whereas collinear or clustered anchors increase it and degrade accuracy (Zekavat & Buehrer 2019, 666-667).

Symbols for formulas 7-9

σ_R = standard deviation of one range measurement (m)

$G = (A^T A)^{-1}$ = geometry (inverse normal) matrix (dimensionless)

c = speed of light (m/s)

$\sigma_x, \sigma_y, \sigma_z, \sigma_t$ = standard deviations of estimated *x, y, z* and clock-bias state (units: m, m, m, s)

A = design/geometry matrix from the linearization (dimensionless)

C_{pp} = covariance matrix of estimated coordinates (m^2)

$HDOP/GDOP$ = geometry metrics; higher geometry amplifies errors more (dimensionless)

Figure 2 illustrates the qualitative meaning of HDOP in two dimensions: when anchors are clustered, the geometry matrix becomes ill-conditioned for large parts of the scene and the resulting HDOP increases, indicating stronger geometric amplification of range noise. (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667.)

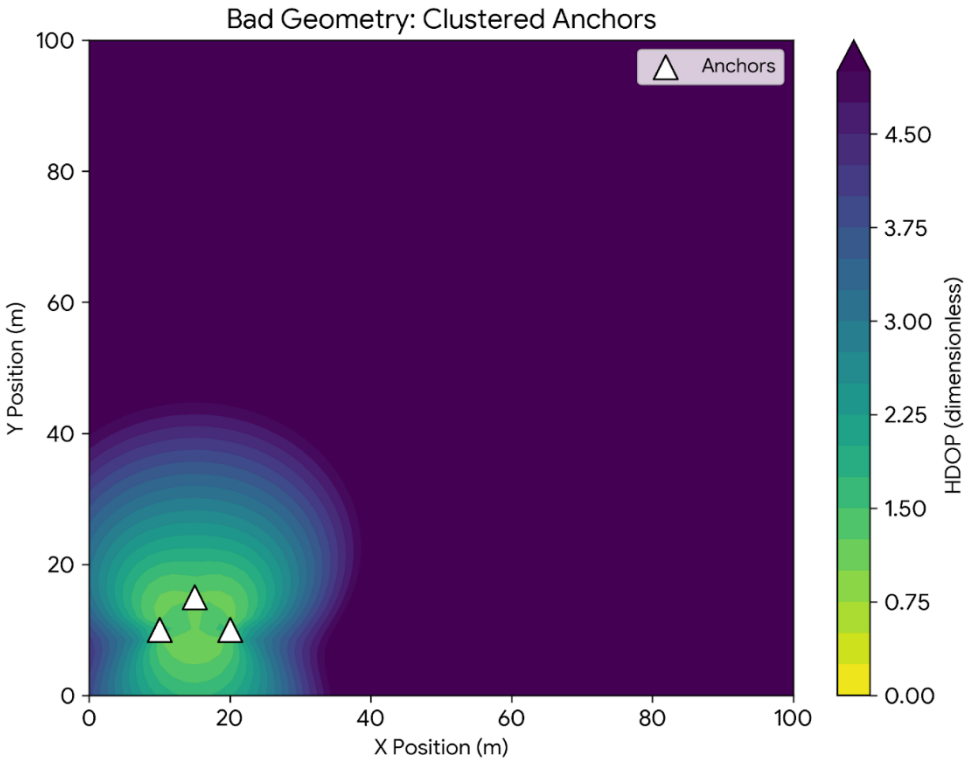

FIGURE 2. Example of poor measurement geometry: clustered anchors produce high HDOP over most of the area (conceptual).

In contrast, Figure 3 shows that widening anchor separation generally reduces HDOP by improving the conditioning of the geometry matrix, thereby reducing the geometric amplification of measurement noise. (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667.)

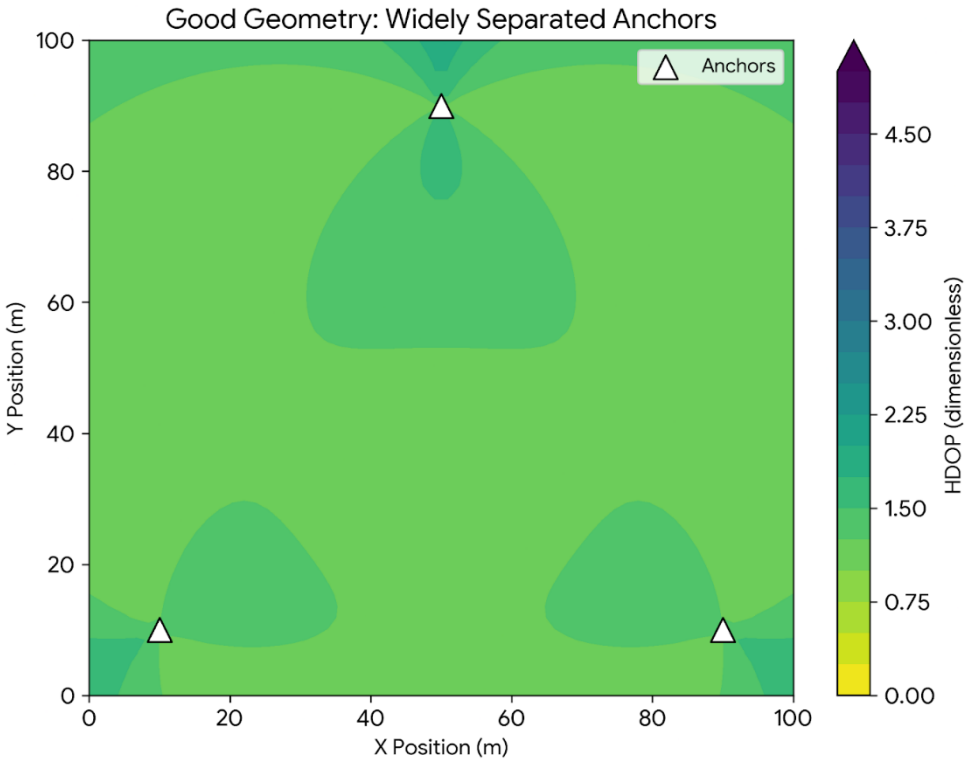

FIGURE 3. Example of improved measurement geometry: widely separated anchors reduce HDOP and improve expected localization robustness (conceptual).

2.1.5 Angle of Arrival (AoA) triangulation

AoA estimates the bearing of a signal rather than its range. With two phase-coherent receive channels and a fixed antenna spacing, a small phase interferometer is formed: the plane wave reaches one antenna slightly earlier than the other, producing a measurable inter-channel phase difference. Bearings collected at successive drone positions can be intersected or fused with the RSSI based ranges from sections 2.1.2 - 2.1.4, to localize the source by triangulation (Bensky 2016, 224-226; Bensky 2008, 200-205).

Figure 4 illustrates the principle stated in Section 2.1.5: AoA provides a bearing estimate that can be intersected across multiple receiver positions to form a location hypothesis. Because bearing estimates have uncertainty, practical geolocation is better interpreted as an intersection region (or an

optimisation/fusion problem) rather than a single exact crossing. (Bensky 2016, 224-226; Bensky 2008, 200-205.)

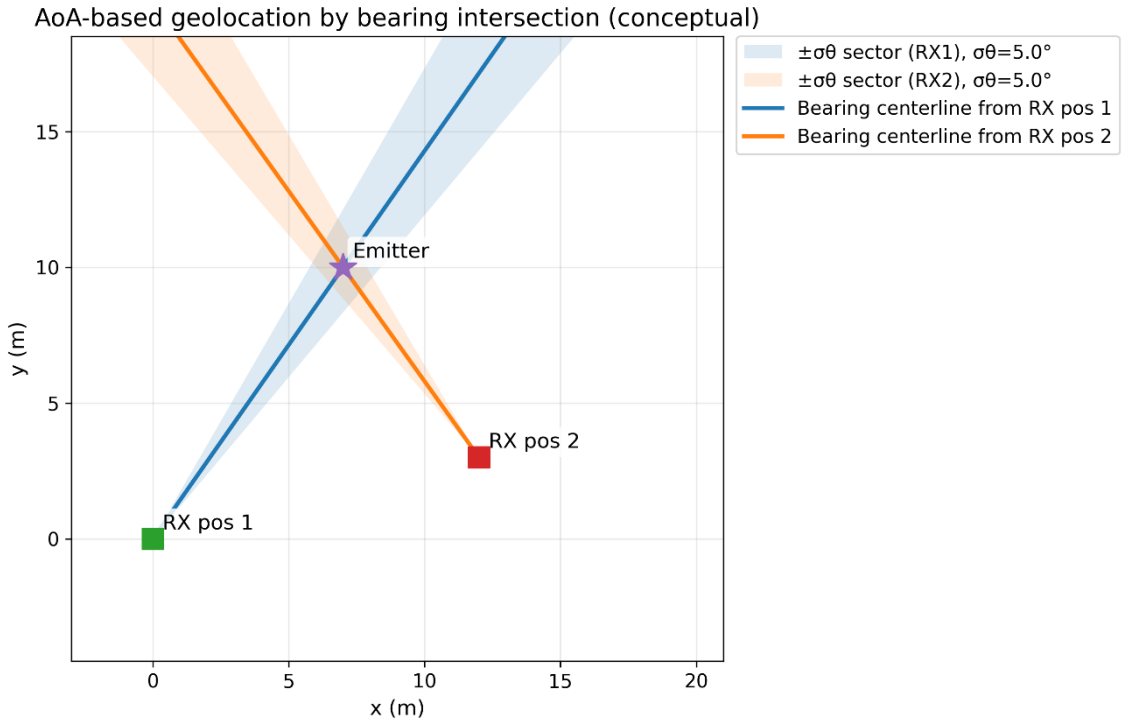

FIGURE 4. AoA-based geolocation by bearing intersection from two receiver positions (conceptual).

Shaded sectors illustrate a bearing uncertainty interval $(\pm\sigma\theta)$, emphasising that practical bearings intersect as a region rather than at a single point.

In the far field, let d be the antenna baseline, λ the wavelength and $\Delta\phi$ the measured inter-channel phase (radians). Defining the normalized spacing $k = d/\lambda$, the bearing θ relative to the baseline normal is given by formula 10 (Bensky 2008, 201-202; Bensky 2016, 224-226).

$$\theta = arcsin\left(\frac{\Delta\phi+2\pi n}{2\pi k}\right)$$

$$n \in \{0, \pm1, \pm2, \dots\},$$

$$k = \frac{d}{\lambda}$$ FORMULA 10

This two-element configuration is unambiguous over $-90° \leq \theta \leq 90°$ when $d \leq \frac{\lambda}{2}(k = 0.5)$ and $|\Delta\phi| \leq \pi$. For the common half-wavelength baseline with no wraps $(d = \frac{\lambda}{2}, n = 0)$, (Formula 10) reduces to formula 11 (Bensky 2008, 201-202; Bensky 2016, 224-226).

$$\theta = \arcsin(\frac{\lambda\Delta\phi}{2\pi d}) \qquad \text{FORMULA 11}$$

Symbols for formulas 10-11

θ = estimated bearing to the source (rad)

$\Delta\phi$ = phase difference between the two RX channels (rad)

d = antenna baseline (m)

λ = wavelength (m)

k = normalized spacing d/λ (dimensionless)

n = ϵ {0, $\pm 1, \pm 2$, ...}(dimensionless)

Practical notes.

Increasing *d* (that is larger *k*) improves small angle sensitivity, but $d > \lambda/2$ introduces wrap ambiguities (multiple θ give the same phase), which can be resolved by a short yaw sweep or by adding a third element at a larger spacing. Multipath can bias $\Delta\phi$ so brief averaging and gentle drone rotation during acquisitions are recommended. Bearings from different drone positions can be intersected to yield a position estimate with an uncertainty ellipse. In real scenes, reflections (ground, buildings, airframe) create multipath, so the measured inter-channel phase $\Delta\phi$ is not purely Line Of Sight (LOS). This can bias the bearing, especially near $|\theta| \to 90°$, and is the main field error source. Mitigating it by a brief yaw sweep (decorrelates reflections) and, if needed, by adding a third element or moving to a new hover point. (Bensky 2016, 228-229; Bensky 2008, 204-205.)

3 HARDWARE PLATFORM OVERVIEW

The measurement platform consists of a software defined radio (Nuand bladeRF 2.0 micro xA9), two tri-band receive antennas (≈700-2600 MHz), low-noise amplifiers with bias-tee feed (one per antenna), a Raspberry Pi 5 host, and a dual-band GNSS module (LC29H) that time-tags each RSSI sample with coordinates. Two phase-coherent receive chains allow both RSSI heat mapping and, when needed, bearing estimation via inter-channel phase (AoA). To preserve sensitivity, each LNA is mounted at the antenna feed; power is injected from the bladeRF's integrated bias-tee and travels up the coax to the LNA, yielding the usual first-stage noise figure advantage in a cascade (Molisch 2011, 39-40)

From section 2.1 the system requirements are: tuning across bands of interest, sufficient instantaneous bandwidth/sample rate to observe tens of MHz channels, and two coherent receivers for phase-based AoA. Section 3.1 describes how the Nuand bladeRF xA9 satisfies these needs.

Section 3.2 will then outline the Raspberry Pi 5 host and its role as the logging/control computer. Section 3.3 discusses the chosen tri-band antennas and mounting/spacing constraints that affect phase interferometer baselines for AoA. Section 3.4 explains the external LNAs (BT-200) and why locating them at the feed and powering them over coax via bladeRF bias-tee is preferable in a low-weight drone setup. (Molisch 2011, 39-40; Nuand 2025c). Section 3.5 covers the LC29H dual-band L1+L5 GNSS module that provides position/time tags for each power sample and supports tighter georeferencing under multipath (Waveshare s.a.).

3.1 Nuand bladeRF 2.0 micro xA9 (SDR)

The bladeRF serves as a two-channel transceiver; in this project it is used as a receiver. It tunes across the bands of interest, down converts ambient signals to complex baseband (I/Q), digitizes them, and streams samples to the host over

USB 3.0. Two phase-coherent receive chains allow both RSSI mapping and, when used, bearing estimates via inter-channel phase (Bensky 2016, 224-229).

How the SDR receive path works (under the hood)

A passive RF waveform collected by the antenna is first amplified at the feed by the LNA. The signal then travels down the coax to the SDR, where a direct-conversion zero-IF or low-IF quadrature mixer translates RF to baseband in two orthogonal paths (I and Q). Low-pass filtering and IQ/DC calibration occur in baseband, after which high-speed ADCs sample I and Q for digital processing and streaming to the host. This is the standard SDR receive chain: quadrature mixing at (near) DC plus digitization "moves" functionality into software while preserving both amplitude and phase for detection/estimation (Tuttlebee 2002, 31-33; 56-66; 99-107).

Device capabilities

The xA9 covers 47 MHz–6 GHz receiver frequency and supports 2 x 2 MIMO reception with 61.44 MS/s complex sampling rate and ≈56 MHz filtered bandwidth (Nuand 2025a). Nuand's 2023.02 update also exposed 122.88 MS/s modes (with reduced bit-depth) for very wide snapshots when needed (Thompson 2023). The xA9 variant integrates a Cyclone V FPGA (~301 kLE) for headroom if future on-radio DSP is desired, and the USB 3.0 interface sustains high-rate I/Q streaming from the airframe (Nuand 2025a). Internally, the platform uses an Analog Devices AD9361 direct-conversion RF transceiver, consistent with the 56 MHz instantaneous (IBW) and the over-clocked 122.88 MS/s mode noted above (Nuand 2025a; Thompson 2023).

Why this SDR

In one compact unit the xA9 provides the wide and fine tuning, phase-coherent dual RX required for AoA, USB 3.0 streaming, and software-switchable +5 V bias-tee so that each LNA can be mounted at the antenna feed yet powered from the radio keeping airborne wiring simple and preserving front-end sensitivity (Nuand 2025a).

The bladeRF is also powered from a PD power bank using PD to DC 5 V cable into the radio's external barrel jack, ensuring stable power for the bias-tee.

3.2 Raspberry Pi 5 Host computer

The Raspberry acts as the on-board processing and logging unit for the measurement system. It configures the SDR, manages storage, and associates every RF power sample with position/time from the GNSS HAT module LC29H.

I/O & throughput

For high-rate SDR work the I/O is decisive: Raspberry Pi provides two USB 3.0 ports supporting simultaneous 5 Gb/s operation, so the bladeRF can stream while an external SSD runs in parallel (Raspberry Pi Ltd 2025).

Power (brief)

Power and stability matter in flight. The Pi 5 is powered from 130 W PD power bank (same for bladeRF) via a PD Power Expansion Board that negotiates PD and delivers 5 V /5 A. This keeps wiring simple and makes Pi 5 suitable for airborne use (Raspberry Pi Ltd 2025).

Timing & headers

For consistent geotagging, the Pi 5 includes a real-time clock (RTC) and the standard 40-pin header, which is used to bring in a 1 PPS signal from the GNSS HAT useful for stable timestamping before/after GPS lock. Raspberry Pi Ltd states the board is expected to be in production until at least January 2036, which reduces supply risk for follow-on work. (Raspberry Pi Ltd 2025.)

3.3 Tri-band antennas

The two tri-band antennas are compact radiators to cover three separated frequency regions with a single physical element, reducing the need for band-specific radiators on space- and weight-constrained platforms such as small unmanned aircraft (Balanis 2016, 846-847).

Specifications and operating principle

Nuand's tri-band antenna is specified for 700-2600 MHz with an SMA-male connector, nominal 5 dBi gain, VSWR ≤ 1.5, 11 inch length, and a flexible hinge; it is sold as a general purpose antenna (Nuand 2025b). Multiband operation in such compact whips relies on deliberate geometry that supports multiple resonances and broadband matching so that one radiator can efficiently couple over separated frequency regions without external switching (Balanis 2016, 846-847).

Why this antenna

This antenna was chosen because its wide coverage aligns with passive survey bands project uses and the bladeRF receiver front end, avoiding antenna swaps while keeping the airframe payload simple. The SMA-male interface mates directly with the BT-200 LNA, minimizing adapters and cost of the project (Nuand 2025b; Nuand 2025c).

3.4 Low-noise amplifiers (LNAs)

LNAs are placed as the first active stage in the receive chain to set the overall system noise figure and to protect weak signals from subsequent losses and receiver noise, which improves link budget and sensitivity (Molisch 2011, 39; Reyland 2024, 24-25). In practice, gain must be balanced against linearity to avoid overload and intermodulation in strong-signal environments (Reyland 2024, 244-246; Molisch 2011, 37-40).

Specifications and operating principle

Nuand's BT-200 bias-tee LNA is a compact in-line module powered over the bladeRF 2.0 RF port and intended for RX only. It uses a Mini-Circuits TSS-53LNB+ MMIC covering ~0.5-5 GHz, provides typical gain of ≈ 20 dB at 200 MHz and ≈ 16 dB at 2.4 GHz, consumes ~300 mW, and falls back to bypass state (~4 dB insertion loss) when unpowered. The bias-tee can be enabled from software, but current draw should be considered if multiple modules are powered from

USB; an external DC input is recommended when needed. (Nuand 2025c.) Functionally, the BT-200 reduces the receiver's equivalent noise figure by placing low-noise gain directly at the antenna port, so later stages contribute proportionally less noise (Molisch 2011, 39; Reyland 2024, 24-25).

Why this LNA

The BT-200 integrates mechanically with the selected SDR and bias-tee control, minimizing cabling and weight on the drone while covering 700-2600 MHz survey bands within its ~0.5-5 GHz LNA passband (Nuand 2025c).

3.5 Dual-band GNSS module (LC29H(DA))

Dual-frequency GNSS receivers track two bands (here L1 ≈ 1575.42 MHz and L5 ≈ 1176.45 MHz) to suppress first-order ionospheric delay and improve robustness in urban multipath, yielding substantially better absolute positioning than single-band designs (Bensky 2016, 257; Zekavat & Buehrer 2019, 676).

Specifications and operating principle

The LC29H(DA) HAT for Raspberry Pi uses a Quectel LC29H series module that tracks L1+L5 and supports concurrent multi-GNSS (GPS, Galileo, BDS, QZSS, plus GLONASS on L1), with on-board LNA/SAW filtering; the DA variant adds Real-Time Kinematic (RTK) rover capability for centimetre-level solutions when corrections are provided. The HAT interfaces via the standard 40-pin GPIO/UART on Raspberry Pi and is shipped with a small active GNSS antenna and IPEX-to-SMA pigtail for external placement, easing integration on the airframe. (Waveshare s.a.)

Why this GNSS module

LC29H(DA) geotags all passive RF measurements with higher accuracy and faster convergence across environments: dual-band L1/L5 mitigates ionospheric error and improves tracking in challenging scenes, and for future updates RTK enables cm-level positioning when correction link is available (Bensky 2016, 257; Waveshare s.a.). The Pi-HAT form factor minimizes cabling and weight and

connects directly to the Raspberry Pi 5 used in logging stack, keeping the GNSS time/position stream tightly synchronized with the receiver data path (Waveshare s.a.).

Figure 5 presents the complete set of receiver prototype components selected and used in this thesis, including the coherent dual-channel SDR, host computer, receive antennas, in-line LNAs, GNSS receiver/antenna, and the associated power and RF interconnects. This overview complements the component-level descriptions in Sections 3.1-3.5 and clarifies the physical between subsystems prior to the integration and software pipeline described in Chapter 4.

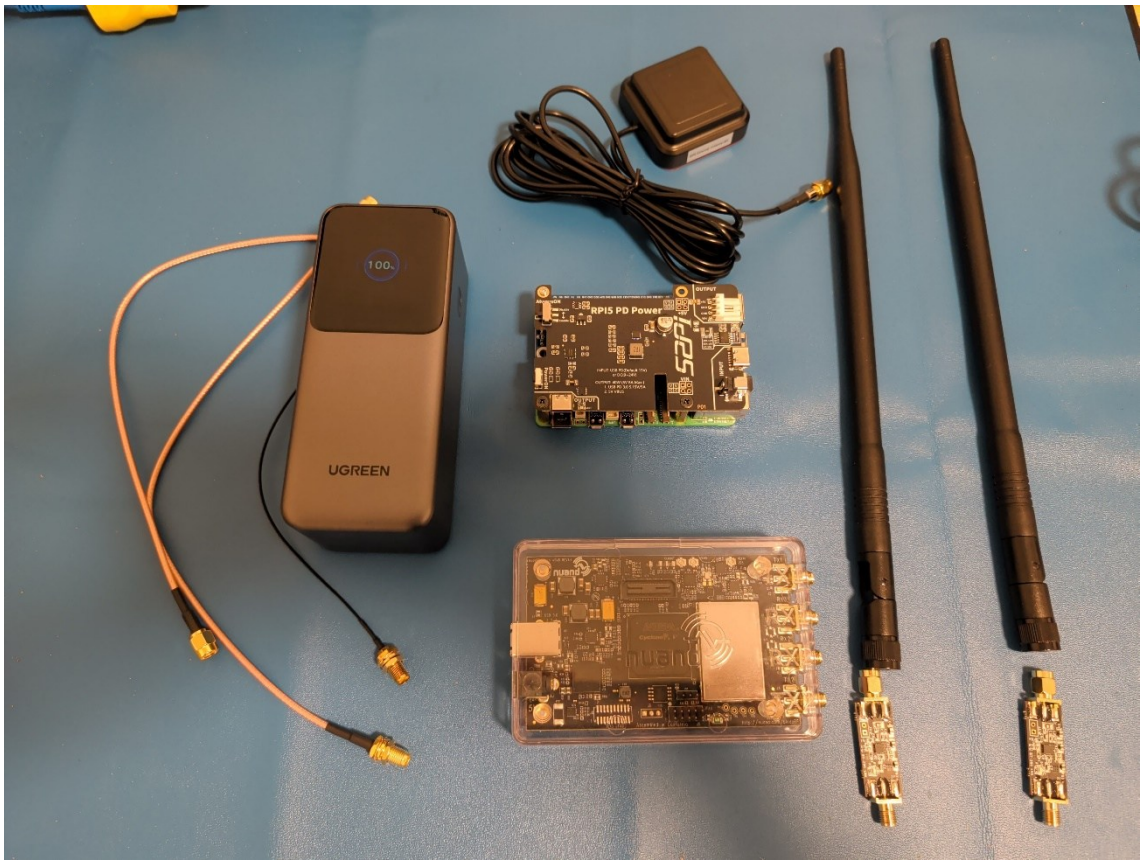

FIGURE 5. Receiver prototype hardware components used in the implementation (laid out before integration).

4 METHODS AND IMPLEMENTATION

This thesis follows an engineering development approach in which a drone-mounted passive RF localization prototype is designed, implemented and evaluated through outdoor test measurements. The method consists of four main phases: (1) deriving system requirements from propagation and localization theory (Chapter 2); (2) selecting and integrating suitable SDR, antenna, LNA and GNSS hardware into a coherent airborne platform (Chapter 3); (3) implementing a data-logging and map-generation pipeline that converts RF measurements into localization products; and (4) applying the prototype in representative test scenarios to assess localization accuracy and feasibility relative to research questions defined in Chapter 1.

4.1 System integration on the drone

The two receive antennas are mounted on lightweight outriggers with a center-to-center of approximately half a wavelength at the operating band to keep the two-element phase interferometer unambiguous over $\pm 90°$, as defined by the AoA spacing condition (Formula 10-11).

Because the antennas are close to the vehicle, nearby conductors (arms/legs, battery, fuselage) act like a finite ground plane that distorts the element pattern and introduces diffracted/scattered fields; measured patterns for monopoles above finite ground planes show clear lobe changes relative to an ideal infinite plane, so physical clearance and symmetry on airframe matter. (Balanis 2016, 703-706.)

In practice the whips are mounted on non-conductive standoffs, keep both elements at equal height, route the two RF leads symmetrically, and avoid running coax tight against metallic arms to reduce coupling and pattern tilt; any residual phase bias from airframe reflections is mitigated in flight by a short, gentle yaw sweep before logging AoA snapshots (Bensky 2008, 204-205; Bensky 2016, 228-229).

Figure 6 shows the assembled receiver prototype used in the outdoor demonstration, highlighting the main interfaces between the SDR front-end, host computer, GNSS receiver, and the two RF receiver chains. The wiring layout reflects the integration intent described in this chapter: phase-coherent dual-channel reception for AoA, combined with GNSS-tagged logging and a compact power arrangement suitable for future unmanned-platform mounting.

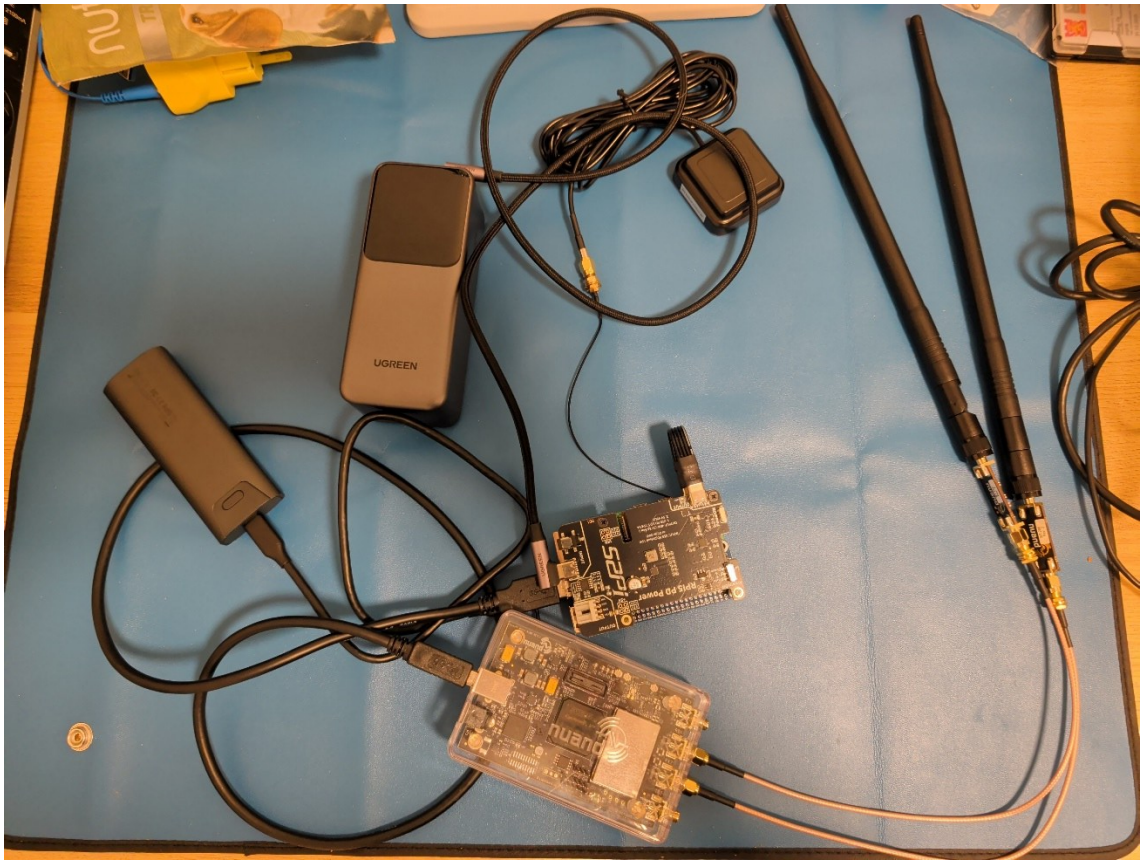

FIGURE 6. Integrated receiver prototype used in field measurements (assembled and interconnected)

4.2 Data acquisition and logging software

The Raspberry configures the bladeRF for dual-channel capture, streams interleaved SC16_Q11 I/Q samples, converts them to floating-point full-scale, forms a windowed FFT, extracts a band- or peak-power estimate in dBFS, and writes each measurement to CSV together with system time fields and GNSS position. The libbladeRF stream uses the synchronous interface with the SC16_Q11 format, where signed integers in (-2048, 2048] represent real values

in (-1.0, 1.0), with I then Q interleaved per channel (Doxygen 2023a, Doxygen 2023b). A condensed implementation of the device setup, dual-RX streaming and per-channel configuration is provided in Appendix 1.1.

Bias-tee control is per channel via `bladerf_set_bias_tee`, applying +5 V to the RX SMA ports (Doxygen 2023c; Doxygen 2023d). The synchronous RX example follows libbladeRF's blocking I/O API and layout RX_X2 for two coherent channels (Doxygen 2023b).

In the implementation, SC16_Q11 sample blocks are converted to complex floating-point I/Q vectors, multiplied by a Hann window, transformed with NumPy's FFT, and converted to magnitude and power spectral density. A helper function computes both the mean power over the selected sub-band and the per-bin power in dBFS for later plotting and inspection, and is listed in Appendix 1.2 (The SciPy community s.a.; NumPy Developers s.a.-a). The Hann taper reduces spectral leakage; the windowed-power factor $U = \frac{1}{N}\sum w^2$ restores energy after windowing, and NumPy's default FFT normalisation is "backward", that is, the forward transform is unscaled (The SciPy community s.a.; NumPy Developers s.a.-a).

After computing per-bin powers, the logger integrates either a channel mask $k \in [k_0, k_1]$ or takes spectral peak and writes a CSV record with UTC, GNSS (φ, λ, h), drone attitude, and dBFS band power for each channel using Python's `csv.DictWriter` (Python Software Foundation 2025).

4.2.1 dBFS definition and scaling in the logger

The logger reports power in decibels relative to full scale (dBFS) so that values are referenced to the converter's full-scale digital range and remain comparable across gain settings; by convention the RMS of a full-scale sine wave is treated as 0 dBFS (Reich s.a.; Arrants, Brannon & Reeder s.a.).

For a complex frame $x[n] \in (-1,1)$ after SC16_11 normalization, the RMS of a full-scale sine equals $1/\sqrt{2}$. Let P_{band} be the integrated linear power over selected

FFT bins with window-power correction *U* and NumPy's "backward" FFT normalization; the logger converts to dBFS as

$$dBFS = 10log10\ (\frac{P_{band}}{\left(\frac{1}{\sqrt{2}}\right)^2}) = 10log10\ (\frac{P_{band}}{0.5}),$$ FORMULA 12

which places a full-scale sine at 0 dBFS by definition (NumPy Developers s.a.-a; Reich s.a.; Arrants, Brannon & Reeder s.a.).

A concise implementation of reading the synchronous RX buffer, de-interleaving SC16-Q11 samples to complex floating-point frames for both channels, and computing band power according to (Formula 12). A condensed implementation of the synchronous sample capture, scaling, FFT, and averaging used for band-power computation is provided in Appendix 1.2.

The use of dBFS makes the measurement independent of unknown absolute gains until a calibration to dBm is performed; conversion to absolute power requires a known front-end gain and impedance and is outside the scope of this passive, relative-strength study (Arrants, Brannon & Reeder s.a.). The CSV writer follows the standard csv library, with `DictWriter` producing a header via `writeheader()` and key-value rows thereafter (Python Software Foundation 2025).

4.2.2 Operational summary

Before flight, GNSS is checked for a valid fix so that each RF record has a reliable timestamp and coordinates; the Raspberry header can supply 1 PPS from the GNSS HAT for stable timing.

Receive gains are set below clipping to avoid intermodulation and bias in power-based metrics (dBFS remains digital, relative unit).

A short live capture verifies that both coherent channels stream, SC16_Q11 is correctly de-interleaved and scaled to [-1,1], and the CSV contains time, GNSS fields, and per-channel dBFS.

When AoA snapshots are planned, a brief hover with a slow yaw sweep is performed at measurement points to reduce multipath bias in inter-channel phase before bearings are computed (Bensky 2016, 224-229).

Absolute dBm calibration is not applied in this prototype; relative dBFS is logged with a fixed transmitter frequency set in the logging script, keeping all FFT/window settings constant for comparability.

These procedures operationalize the chosen development method by providing a repeatable way to acquire georeferenced RSSI and AoA data needed to answer questions on localization accuracy, geometry effects and AoA feasibility (Chapter 1).

4.3 Heat-map generation (CSV to map)

CSV produced by the logger is transformed into an interactive map, and the script provides qualitative localization cues from received-power features and optional bearing overlays.

4.3.1 Inputs and parameters for RSSI heat mapping

This subsection defines the required inputs and the parameters that affect only the heat-layer rendering. The map is constructed from geotagged rows (`gps_lat, gps_lon`) and the one power metric (RSSI% or dBFS), which are converted into unit-interval weights before rendering the layer. The expectation that higher received power tends to occur nearer the emitter follows the free-space/log-distance baseline introduced in Chapter 2. The specific normalization and defaults reflect the map-making script's `argparse` configuration. (Molisch 2011, 48-49; Bensky 2008, 140-146.)

The map-making workflow, summarized as a UML micro-workflow diagram in Appendix 4, first chooses and scales the signal columns to weights (for example, min-max scaling for dBFS), optionally filters out weak points, and then passes [lat, lon, weight] triples to the Leaflet heat-map plugin. Stronger received power

is expected closer to the emitter under the log-distance model, so higher weights visually emphasize likely source regions (Bensky 2008, 144-146; Molisch 2011, 48-49).

From a software standpoint, this is implemented by the Folium HeatMap plugin receiving pre-computed weights and NumPy vectorization for the list construction (Folium s.a.; NumPy Developers s.a.-b). A compact example of the map-layer creation routine is included in Appendix 2.1 and Appendix 2.4-2.5.

The full set of command-line parameters and defaults for the map-making script, including I/O options, extent and framing, heat-layer radius and blur, weighting, AoA overlay, and grid-aggregation settings, is summarized in Appendix 3.

4.3.2 Optional AoA bearings: what they are and how they appear on the map

When the CSV contains a per-row bearing (`bearing_deg`, degrees true), the map overlays short rays pointing toward the hypothesized source from each GNSS fix. The endpoint is computed on WGS-84 via a forward geodesic using pyproj.Geod.fwd(…), and the resulting two-point segment is then rendered as a thin polyline with Folium's PolyLine, providing an immediate directional cue on top of the RSSI heat layer (pyproj s.a.; Folium s.a.).

Code sample 1. The actual overlay is produced by projecting a short geodesic from the fix along the bearing and drawing a segment.

```
lon2, lat2, _ = geod.fwd(float(lon), float(lat), float(brg), float(maxlen_m))

folium.PolyLine([(lat, lon), (lat2, lon2)], weight=2, opacity=0.55,
tooltip=f"AoA {brg:.1f}°").add_to(m)
```

This uses a WGS-84 geodesic to compute a point `maxlen_m` meters along the azimuth and renders a short "direction-to-source" ray from each measurement location, providing an immediate directional cue on top of the RSSI heat (Bensky 2016, 224-226). In practice the endpoint comes from the forward geodetic calculation and the segment is drawn with Folium's polyline (Folium s.a.).

A complete implementation of the AoA overlay, including map initialization and layer management, is provided in Appendix 2.5-2.6.

Code sample 2. When phase and baseline/frequency are available, the script can derive the AoA and write it as a column for later plotting:

```
phi = pd.to_numeric(df[phi_col], errors='coerce').to_numpy()

phi = np.arctan2(np.sin(phi), np.cos(phi)) # wrap to [-pi,pi]

if np.isscalar(lam):

arg = (lam * phi) / (2.0*np.pi*baseline)

else:

lamv = lam.to_numpy()

arg = (lamv * phi) / (2.0*np.pi*baseline)

arg = np.clip(arg, -1.0, 1.0)

rel_deg = np.degrees(np.arcsin(arg))

if heading_col and heading_col in df.columns:

hdg = pd.to_numeric(df[heading_col], errors='coerce').to_numpy()

bear = (hdg + rel_deg) % 360.0

else:

bear = (rel_deg + 360.0) % 360.0

out = df.copy()

out['bearing_deg'] = bear
```

This encapsulates the interferometer arcsine relation and converts the relative angle to a true-north bearing for mapping (Bensky 2008, 201-202; Bensky 2016, 224-226). The computation uses standard NumPy trigonometric functions with wrap-safe phase handling and an optional heading offset; the resulting `bearing_deg` column is written back to the table for plotting (NumPy Developers

s.a.-b). A more complete AoA computation and CSV integration pipeline is presented in Appendix 2.2.

4.3.3 Map output and qualitative localization cues

The map-making stage converts the logger CSV into an interactive HTML output that supports qualitative localization cues rather than automatic point estimate. The implemented pipeline selects one power-derived metric per row (for example, dBFS) and converts It to unit-interval weights using configurable scaling and shaping, after which low-weight points may be filtered prior to rendering the heat layer (Appendix 2.4 and Appendix 2.5).

To improve visual stability when the receiver path contains stops or slow-motion segments, the script optionally aggregates samples onto a local meter-grid before heat-layer construction. This aggregation is performed in a local azimuthal-equidistant (AEQD) metric frame defined at the mean latitude/longitude of the dataset, after which samples are binned into grid cells and reduced to one representative value per cell using a selectable statistic, thereby reducing the dominance of the oversampled segments in the heat layer (Appendix 2.3).

The HTML output is composed as a layered map content. The route polyline is drawn from the GNSS track (optionally sub-sampled), while the heat layer built from the (optionally aggregated) latitude, longitude, weight triples. To improve inspectability, the strongest weighted cells may be marked with tooltips that report the underlying metric value and the number of samples contributing to the cell (Appendix 2.5).

If a bearing column is present and AoA overlay is enabled, short rays are rendered from each plotted position along the bearing direction. The ray endpoint is computed by a forward geodesic on WGS-84, after which a two-point polyline is drawn on top of the heat layer to provide a directional cue that complements the power-based “hot region” visualization (Appendix 2.6).

The field evaluation of these outputs, including the calibration procedure and the resulting map interpretation for the drive-by demonstration, is presented in Chapter 5.

4.3.4 Assumptions and limitations (interpretation of the map)

This mapping setup assumes that higher received power tends to occur nearer to the emitter according to the log-distance model; log-normal shadowing typical spreads of 5-8 dB means individual samples can deviate substantially, so the heat layer conveys trends rather than precise iso range contours (Bensky 2008, 144-146; Haslett 2008, 176-178).

The Leaflet heat layer integrates both per-row weights and sampling density; therefore, dense flight paths can visually outweigh sparser traversals even at the same average power, unless –min-weight filtering or pre-aggregation is used (Folium s.a.)

Absolute power is not available unless a calibration constant or table is applied; dBFS values are relative to full-scale and require a known gain/impedance to convert to dBm, which the prototype does not apply by design at this stage (Arrants, Brannon & Reeder s.a.).

When present, AoA features are computed from a two-element phase interferometer; bearings are sensitive to multipath and to airframe-induced pattern/phase biases, so brief hovering/yaw averaging and adequate baseline geometry are recommended (Bensky 2016, 224-229; Balanis 2016, 703-706).

For triangulation and fusion, estimator spreads depend strongly on vantage geometry as captured by DOP/HDOP: widely separated looks reduce geometric amplification of noise, whereas clustered or nearly collinear looks increase it (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667).

For grid fusion, (AEQD) projection around the scene is used to ensure near-metric behavior over the small area of interest; the RSSI cost uses the log-

distance model and a per-grid intercept, and the angular cost penalizes bearing residuals with an assumed 1σ width (Bensky 2008, 144-146).

4.4 System integration for AoA

The two receive antennas are installed on short outriggers with a center-to-center baseline of approximately half a wavelength at the operating frequency to keep two-element phase interferometer unambiguous over $\pm 90°$ field and to avoid angular aliasing; this half wavelength choice also aligns with classic array-spacing limits that preclude grating lobes when scanning near broadside (Bensky 2016, 224-226; Balanis 2016, 298-299). The same half wavelength “critical spacing” appears throughout array theory as the angular Nyquist rate, ensuring that phase wraps do not masquerade as additional bearings (Balanis 2016, 967-969).

When higher bearing precision is required, increasing the relative element separation $k = d/\lambda$ steepens the phase-versus-angle slope and reduces estimator variance, but this also introduces 2π phase ambiguities; a pragmatic remedy is a mixed-baseline layout (example, one half wavelength pair for coverage and one longer pair for accuracy) or time-multiplexing to a third element, resolving the coarse sector with the short baseline and refining with the long one (Bensky 2016, 227-229; Balanis 2016, 324-325).

On the airframe, both elements are mounted symmetrically and as far as practical from conductive structures as that each “sees” a similar environment; nearby metal modifies the active element patterns via edge diffraction from finite ground planes and related scattering, which can bias the inter-element phase and thus the bearing estimate (Balanis 2016, 703-706). Short, rigid booms keep the baseline fixed under vibration, and clear line-of-sight to the emitter is favored to reduce multipath-induced phase errors (Bensky 2016, 224-226); mutual coupling between closely spaced elements further alters the active patterns and impedances (Balanis 2016, 474)

5 FIELD TESTING AND RESULTS: CONTINUOUS-WAVE DRIVE-BY DEMONSTRATION

This chapter reports the outdoor field demonstration conducted as a car-based drive-by test using a fixed continuous-wave (CW) transmitter near 2.5 GHz and a dual-channel coherent receiver setup. The objective is to evaluate the end-to-end prototype workflow under realistic operating conditions, from calibration and logging to map generation and qualitative direction-of-arrival interpretation. The chapter first describes the test environment, instrumentation, and measurement parameters, then documents the calibration procedure used to compensate constant inter-channel phase bias, and finally presents the resulting heatmap and AoA ray overlay together with an interpretation of observed behavior and limitations.

5.1 Test objective and evaluation approach

The evaluation focused on functional validation under representative outdoor propagation rather than on reporting absolute localization accuracy in meters. A controlled narrowband reference signal was selected to support stable phase-based interferometry and to allow a short stationary calibration capture that estimates the constant inter-channel phase bias. The qualitative success criteria were defined as follows: (1) the RSSI/peak-dBFS heat layer should exhibit a distinct high-intensity region near the known transmitter location under the log's strongest received levels; (2) the AoA rays plotted from measurement points should predominantly orient toward the known transmitter area when AoA gating indicates high coherence and adequate SNR; and (3) the results should remain interpretable when plotted as an interactive HTML map using the implemented map-making pipeline. (Appendix 2; Bensky 2016, 224-226.)

5.2 Test setup and instrumentation

5.2.1 Transmitter configuration

Transmitter was a Rohde & Schwarz vector signal generator configured to produce a single-tone continuous-wave signal (CW) near 2.5 GHz, with the transmitter fixed at a known location during the run. CW denotes an unmodulated carrier that is spectrally narrow and phase-stable, making it suitable for phase-calibration and interferometric AoA estimation. The generator output level was set to 0 dBm during the demonstration.

5.2.2 Receiver platform and logging chain

The receiver platform consists of a coherent dual-channel SDR front-end operating in synchronized dual-receiver mode and a GNSS receiver providing latitude and longitude for each logged measurement row. During data capture, the logger computes band- and peak-power metrics, a local SNR estimate around the detected spectral peak, and a coherence metric derived from the complex relationship between the two receiver channels. When quality-gating conditions are satisfied (coherence and SNR thresholds, together with clipping and validity checks), the logger records the inter-channel phase difference at the detected peak and converts the phase difference to an AoA estimate using the two-element phase interferometer model described in Chapter 2 (Bensky 2016, 225-227).

The dataset produced by the demonstration is stored as a CSV file that includes system timestamp fields, GNSS coordinates, RF metrics (band/peak values), SNR, coherence, inter-channel phase difference, AoA output, and auxiliary fields required for post-processing (for example, baseline and frequency parameters used for the AoA mapping).

5.2.3 Antenna baseline selection and handling during demonstration

AoA estimation in a two-element phase interferometer maps the measured phase difference to an arrival angle using the antenna separation d (baseline) and the signal wavelength λ (Bensky 2016, 225-227). The baseline is constrained by ambiguity considerations: for a given baseline, the unambiguous angular span decreases as d increases, because multiple incident angles can produce phase differences that are indistinguishable after phase wrapping (Bensky 2016, 225-227). For the 2.500 GHz demonstration, the antenna baseline was selected as 0.085 m and supplied explicitly to the logger. The baseline selection follows the two-element phase interferometer relationship and the ambiguity considerations presented in Section 2.1.5 (Formulas 10-11).

During the calibration capture and the drive-by capture, the two receive antennas were held manually with approximately the intended separation. This handling method is sufficient for feasibility demonstration, but it introduces additional uncertainty because small baseline changes and unobserved rotations can occur during motion. Consequently, part of the observed AoA spread in the visualization is attributable to practical handling variation in addition to propagation effects.

5.2.4 Calibration capture and application of constant phase offset

Prior to the drive-by run, a short stationary calibration capture (approximately 10 s) was recorded under a strong coherent received signal. The purpose of calibration is to estimate a constant inter-channel phase bias caused by hardware and RF path differences between the two receiver channels. Such channel-to-channel biases are a known practical issue in multi-antenna measurements and can distort direction-finding results unless calibrated (Molisch 2011, 161-162). In addition, direction-of-arrival methods are sensitive to array calibration assumptions (for example, channel gain/phase alignment and element responses), and calibration errors can become a dominant error source even when the received signal itself is strong (Molisch 2011, 162).

In the demonstrated workflow, the calibration phase offset `phase_cal_rad` is estimated offline from the calibration CSV by filtering phase samples using coherence/SNR thresholds and then computing a circular mean. The resulting constant phase offset is then applied in the logger for the drive-by run by subtracting the calibrated offset from each measured inter-channel phase difference prior to AoA conversion. This step is required because an uncompensated constant phase bias rotates the AoA estimates by a systematic offset, producing rays that are consistently shifted even when coherence remains high.

5.2.5 Heatmap generation and AoA-ray overlay

After logging, the post-processing pipeline reads the GNSS-tagged CSV and generates an interactive HTML map that includes the driven route overlay, a heat layer derived from the selected RSSI metric, and AoA rays drawn for measurement rows that satisfy AoA validity conditions. The AoA overlay is rendered as map bearings by combining the relative AoA estimate with a platform heading estimate. The heading is derived from successive GNSS fixes and is therefore most reliable during continuous motion; when stationary or at very low speed, the heading estimate becomes unstable, and AoA ray rendering should be interpreted with caution.

5.3 Results and interpretation

Figure 7 presents the resulting heatmap and AoA-ray overlay from the drive-by demonstration. The heat intensity concentrates in the area where the receiver passes closest to the fixed transmitter, which is consistent with the expected relationship between received power and distance in outdoor line-of-sight or partial line-of-sight conditions. The AoA rays cluster predominantly toward the known transmitter location, indicating that the calibrated inter-channel phase and the AoA gating strategy provide bearings that are qualitatively consistent with the ground-truth transmitter placement.

The AoA rays exhibit visible angular spread rather than converging to a single bearing. This spread is expected in practice for three primary reasons. First, multipath propagation introduces additional reflected components that perturb the measured phase difference and can vary with receiver position, which is a fundamental characteristic of mobile radio channels (Molisch 2011, 28-29). Second, the hand-held antenna baseline introduces mechanical variability and unobserved orientation changes, which directly perturb the interferometer geometry used for phase-to-angle mapping. Third, residual calibration error and time variation in the RF front-end (for example, temperature drift) can introduce small phase deviations that accumulate into angular dispersion.

TABLE 1. Top 5 valid AoA samples from the test run CSV.

gps_lat	gps_lon	peak_dBFS_rx0	peak_dBFS_rx1	snr_db	coh	phase_diff_rad
60,75572433	24,80329047	-22,87732	-22,65025	42,445311	0,99929148	-2,464356
60,75569345	24,80319675	-18,08223	-20,13625	46,331464	0,99986023	-2,379797
60,75569345	24,80319675	-17,47458	-21,24898	45,300619	0,9995532	-2,291121
60,75569345	24,80319675	-17,49562	-24,08677	47,037517	0,99979889	-2,649627
60,75569345	24,80319675	-18,8806	-27,22068	43,943809	0,99923277	-2,726919

Figure 7 visualises the drive-by result as a combined route, heat layer, and AoA overlay. The blue polyline indicates the GNSS-derived receiver route during the pass. The heat layer encodes the selected received-level metric (peak dBFS) such that higher intensity bearings plotted from selected measurement points.

The red marker indicates the known transmitter location and was added manually for reference.

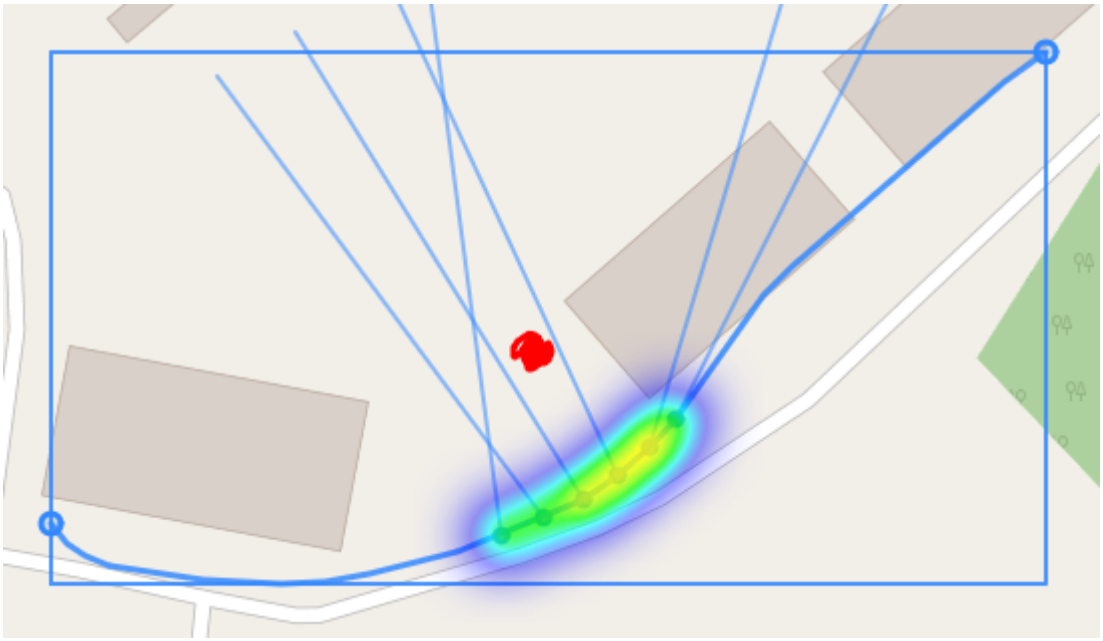

FIGURE 7. Present the resulting heatmap and AoA-ray overlay from the drive-by demonstration.

6 DISCUSSION

This chapter synthesizes the outcomes of the outdoor drive-by demonstration and relates the observed behavior to the research questions and the theoretical basis presented earlier. The discussion emphasizes practical feasibility, interpretation of the qualitative localization products, and development priorities required for transferring the prototype from a ground-based demonstration to a drone-borne system.

6.1 Summary of key findings

The demonstration validated that the implemented end-to-end processing chain coherent dual-channel reception, calibrated inter-channel phase, AoA quality gating, GNSS-tagged logging, and map-based visualization can produce interpretable localization cues under outdoor conditions using a fixed narrowband reference source.

The strongest received levels concentrate spatially in the vicinity where the receiver passes closest to the transmitter, which is consistent with the qualitative expectation that higher received power tends to occur nearer the emitter under log-distance and shadowing-based path-loss models. (Bensky 2008, 144-146; Molisch 2011, 48-49.)

AoA rays plotted from measurement points with high coherence and adequate SNR predominantly orient toward the known transmitter area, indicating that the applied phase calibration and AoA gating strategy are sufficient for producing qualitatively correct bearings in the demonstrated conditions. (Bensky 2016, 224-227.)

6.2 Discussion relative to the research questions

RQ1 (RSSI ranging and trilateration accuracy)

The thesis develops RSSI-to-range conversion and trilateration models and discusses their expected uncertainty sources, including log-normal shadowing and calibration dependence. (Bensky 2008, 140-146; Haslett 2008, 176-178.)

However, the conducted demonstration prioritized functional validation and qualitative map interpretability rather than reporting absolute localization accuracy in meters, and the produced outputs are therefore best interpreted as feasibility evidence for the receiver and mapping pipeline rather than a quantitative RSSI-trilateration accuracy evaluation.

RQ2 (geometry effects via DOP-like metrics)

geometry affects uncertainty through the inverse normal matrix and DOP-style quantities, and it explains why clustered or near-collinear measurement geometries amplify uncertainty. (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667.)

In the demonstrated drive-by, the route geometry provides a single-pass set of measurement points and therefore supports qualitative interpretation (heat concentration and bearing consistency) more directly than a controlled multi-anchor geometry study; consequently, DOP-style assessment remains primarily a methodological tool for planning and evaluating future multi-pass surveys rather than a fully quantified result of the present dataset

RQ3 (feasibility of two-element AoA on an unmanned platform)

The drive-by results support the feasibility aspect of two-element phase-interferometric AoA when coherent reception, baseline selection, calibration, and gating are applied as described. (Bensky 2016, 224-229; Balanis 2016, 703-706.)

At the same time, the results also illustrate that AoA spread is expected in realistic environments due to multipath and implementation factors, and therefore “reliable

AoA” should be interpreted as a quality-controlled bearing cue rather than a single deterministic line. (Molisch 2011, 28-29.)

6.3 Why calibration was decisive for AoA interpretability

The implemented workflow estimates a constant inter-channel phase offset from a short stationary calibration capture and applies the resulting offset during the drive-by run to reduce systematic AoA rotation.

This step is consistent with the broader observation that multi-antenna direction-finding performance can be dominated by channel mismatches and calibration errors even when the received signal itself is strong, because phase- and response-alignment assumptions are integral to array processing. (Molisch 2011, 161-162.)

6.4 Limitations and repeatability considerations

The main limitations arise from the use of a car-based platform rather than a drone, the use of a controlled CW reference rather than bursty handset-like emissions, and hand-held antenna baseline and orientation variation during motion.

The observed AoA spread is consistent with expected mobile-channel behavior under multipath, and it is further increased by baseline/orientation variability and residual front-end drift, which perturb the phase-to-angle mapping used by a two-element interferometer. (Molisch 2011, 28-29; Bensky 2016, 225-227.)

Furthermore, the heading used to rotate relative AoA into map bearings is derived from successive GNSS fixes and is therefore most reliable during continuous motion; when the platform is stationary or moving slowly, heading becomes unstable and AoA rays require cautious interpretation.

6.5 Non-breaking development priorities

The following improvements are consistent with the implemented method and are unlikely to require algorithm redesign:

- Rigid antenna baseline fixture: A mechanically fixed baseline and stable orientation reduce geometry uncertainty and improve repeatability of phase-interferometric AoA. (Bensky 2016, 225-227.)
- Repeated calibration and drift checking: Short calibration captures before and after a measurement run provide evidence for phase stability and support detection of drift-driven bearing bias. (Molisch 2011, 161-162.)
- Improved timing/robustness in GNSS tagging: More explicit handling of GNSS fix quality and timestamp alignment strengthens the interpretability of map overlays and supports more controlled comparisons between passes.
- Planned multi-pass surveys with geometry assessment: Repeating passes around the suspected source area supports DOP-informed geometry improvement and enables stronger evaluation of RSSI-based trilateration and uncertainty propagation in later experiments. (Bensky 2016, 182-184; Zekavat & Buehrer 2019, 666-667.)

6.5.1 Lessons learned (what worked and why)

The implementation confirms Python-based workflow is effective for rapid prototyping of SDR logging, signal processing, and mapping, because prototype integrates coherent streaming, FFT-based power estimation, CSV logging, and HTML map generation within a single iterative development loop. In particular, the combination of numerical libraries for spectral estimation and the mapping pipeline for qualitative interpretation reduced the turnaround time between field captures and analysis, which supported debugging of coherence gating and phase calibration logic.

6.5.2 Lessons learned (what did not work and why)

The drive-by demonstration indicates the mechanical stability is at least as important as signal-processing correctness for two-element interferometric AoA, because baseline variation and unobserved orientation changes directly perturb the phase-to-angle mapping and increase bearing spread even when coherence remains high.

Furthermore, GNSS-derived heading is reliable mainly during continuous motion; at low speed or during stops the heading estimate becomes unstable, which can rotate plotted rays and reduce interpretability of bearing overlays.

If future drone deployment introduces tighter real-time constraints (for example, higher sample rates, stricter timing alignment, or longer duration logging under limited compute), parts of the processing chain may benefit from migration to a compiled implementation; however, the present prototype demonstrates that Python is adequate for functional validation and field-test iteration of the chosen methods.

6.6 Concluding remarks

Overall, the conducted demonstration indicates that the implemented receiver and post-processing pipeline can deliver practically interpretable localization cues RSSI heat concentration and phase-calibrated AoA bearings under outdoor conditions when coherent dual-channel capture and calibration are applied.

The results support the feasibility of the approach as a field-operable prototype while also highlighting that robust deployment on an unmanned platform requires mechanical stabilization, systematic calibration practice, and measurement geometry planning to manage the dominant real-world error sources identified in the literature and observed in the demonstration. (Molisch 2011, 28-29, 161-162; Bensky 2016, 224-229.)

REFERENCES

Arrants, A., Brannon, B. & Reeder, B. s.a. AN-835: Understanding High Speed ADC Testing and Evaluation. Analog Devices, Inc. Available at: https://www.analog.com/en/resources/app-notes/an-835.html. Accessed: 27 August 2025.

Balanis, C. A. 2016. Antenna Theory: Analysis and Design. Fourth edition. Hoboken, New Jersey: John Wiley & Sons, Inc.

Bensky, A. 2008. Wireless Positioning Technologies and Applications. Norwood, MA: Artech House.

Bensky, A. 2016. Wireless Positioning Technologies and Applications. Second edition. Norwood, MA: Artech House.

Doxygen 15 February 2023a. Synchronous API. libbladeRF 2.5.0 documentation. Available at: https://www.nuand.com/libbladeRF-doc/v2.5.0/group__fn__streaming__sync.html. Accessed: 27 August 2025.

Doxygen 15 February 2023b. bladeRF Synchronous Interface: Basic Usage Without Metadata. libbladeRF 2.5.0 documentation. Available at: https://www.nuand.com/libbladeRF-doc/v2.5.0/sync_no_meta.html. Accessed: 27 August 2025.

Doxygen 15 February 2023c. Bias Tee Control. libbladeRF 2.5.0 documentation. Available at: https://www.nuand.com/libbladeRF-doc/v2.5.0/group__fn__bladerf2__bias__tee.html. Accessed: 27 August 2025.

Doxygen 2023d. libbladeRF 2.0 Release Notes. libbladeRF 2.5.0 documentation. Available at: https://www.nuand.com/libbladeRF-doc/v2.5.0/relnotes_2_0.html. Accessed: 27 August 2025.

Folium s.a. Heatmap. Available at: https://python-visualization.github.io/folium/latest/user_guide/plugins/heatmap.html. Accessed: 7 November 2025.

Haslett, C. 2008. Essentials of Radio Wave Propagation. New York: Cambridge University Press.

Keski-Korpela, N. 2025. Katso, miten poliisin drooni nousee laatikosta ilmaan hetkessä – kokeilu alkaa Hervannassa. Yle. Available at: https://yle.fi/a/74-20193344. Accessed: 14 November 2025.

Molisch, A. 2011. Wireless Communications. Second edition. Chichester: John Wiley & Sons, Ltd.

NumPy Developers s.a.-a Discrete Fourier Transform. Available at: https://numpy.org/doc/2.1/reference/routines.fft.html. Accessed: 27 August 2025.

NumPy Developers s.a.-b. Mathematical functions. Available at: https://numpy.org/doc/2.3/reference/routines.math.html. Accessed: 7 November 2025.

Nuand 2025a. bladeRF 2.0 micro. Available at: https://www.nuand.com/bladerf-2-0-micro/. Accessed: 20 August 2025.

Nuand 2025b. Tri-Band Antenna. Available at: https://www.nuand.com/product/tri-band-antenna/. Accessed: 25 August 2025.

Nuand 2025c. BT-200 Bias-tee Low Noise Amplifier. Available at: https://www.nuand.com/product/bt-200/. Accessed: 25 August 2025.

Python Software Foundation. 7 November 2025. csv - CSV File Reading and Writing. Available at: https://docs.python.org/3/library/csv.html. Accessed: 11 November 2025.

pyproj s.a. Geod. pyproj 3.7.2 documentation. Available at: https://pyproj4.github.io/pyproj/stable/api/geod.html. Accessed: 7 November 2025.

Raspberry Pi Ltd 2025. Raspberry Pi 5. Product brief. Available at: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf. Accessed: 20 August 2025.

Reich, R. s.a. AN-938: Digital and Analog Measurement Units for Digital CMOS Microphone Preamplifier ASICs. Analog Devices, Inc. Available at: https://www.analog.com/en/resources/app-notes/an-938.html. Accessed: 27 September 2025.

Reyland, M. J. 2024. Software Defined Radio: Theory and Practice. Norwood, MA: Artech House.

The SciPy community. s.a. scipy.signal.windows. hann. Available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.windows.hann.html. Accessed: 27 August 2025.

Thompson, R. 15 February 2023. 2023.02 Release – 122.88 MHz Instantaneous Bandwidth. Nuand. Available at: https://www.nuand.com/2023-02-release-122-88mhz-bandwidth/. Accessed: 20 August 2025.

Tuttlebee, W. 2002. Software Defined Radio: Enabling Technologies. Chichester: John Wiley & Sons, Ltd.

Waveshare s.a. LC29(XX) GPS/RTK HAT. Available at: https://www.waveshare.com/wiki/LC29H%28XX%29_GPS/RTK_HAT. Accessed: 25 August 2025.

Yle News 2024. Body of 12-year-old avalanche victim recovered in Finnish Lapland. Yle News. Available at: https://yle.fi/a/74-20067791. Accessed: 14 November 2025.

Zekavat, S. A. & Buehrer, M. R. 2019. Handbook of Position Location: Theory, Practice and Advances. Second edition. Hoboken, New Jersey: John Wiley & Sons, Inc.

APPENDICES

Appendix 1 Core logger code

Appendix 2 Map-generation code

Appendix 3 Map-making parameters and defaults

Appendix 4 UML micro-workflow for map-making (CSV to HTML)

Appendix 1.1 bladeRF setup and dual-RX synchronous streaming configuration

```
dev = bladerf.BladeRF()

rx0 = bladerf.CHANNEL_RX(0)

rx1 = bladerf.CHANNEL_RX(1)



for ch, gain, name in [(rx0, gain0, "RX0"), (rx1, gain1, "RX1")]:

try:

dev.set_bias_tee(ch, bool(bias_tee))

except Exception:

pass

dev.set_sample_rate(ch, sr)

dev.set_bandwidth(ch, bw)

dev.set_frequency(ch, tune_hz)

dev.set_gain(ch, gain)

try:

actual = dev.get_gain(ch)

print(f"[SDR] {name} gain set {gain} -> readback {actual}")

except Exception:

pass

dev.enable_module(ch, True)
```

```
dev.sync_config(
 _bl.ChannelLayout.RX_X2,
 _bl.Format.SC16_Q11,
 num_buffers=32,
 buffer_size=n,
 num_transfers=16,
 stream_timeout=3500
 )
```

Appendix 1.2 Synchronous sample capture, scaling, FFT, and averaging

```
for _ in range(avg_frames):
 buf = np.empty(n * 4, dtype=np.int16) # holds 2 channels: (I0,Q0,I1,Q1)*n
(i.e., 2*n complex samples total)
 dev.sync_rx(buf, n_samples_sync, 3500)
 frame = buf.reshape(n, 4)
 i0 = frame[:, 0]; q0 = frame[:, 1]
 i1 = frame[:, 2]; q1 = frame[:, 3]


 raw_max0 = int(np.max(np.maximum(np.abs(i0), np.abs(q0))))
 raw_max1 = int(np.max(np.maximum(np.abs(i1), np.abs(q1))))
 raw_max = max(raw_max0, raw_max1)


 if ADAPTIVE_RAW_SCALE and raw_max > RAW_MAX_SWITCH:
 s = float(INT16_SCALE)
```

```
clip_code = int(INT16_CLIP_CODE)

 else:

 s = float(Q11_SCALE)

 clip_code = int(Q11_CLIP_CODE)



 clip0_cnt += int(np.count_nonzero((np.abs(i0) >= clip_code) | (np.abs(q0) >=
clip_code)))

 clip1_cnt += int(np.count_nonzero((np.abs(i1) >= clip_code) | (np.abs(q1) >=
clip_code)))

 total += n



 iq0 = (i0.astype(np.float32) / s) + 1j * (q0.astype(np.float32) / s)

 iq1 = (i1.astype(np.float32) / s) + 1j * (q1.astype(np.float32) / s)



 if DEFAULT_DC_REMOVE:

 iq0 = iq0 - np.mean(iq0)

 iq1 = iq1 - np.mean(iq1)



 X0 = np.fft.fftshift(np.fft.fft(iq0 * window, n=n))

 X1 = np.fft.fftshift(np.fft.fft(iq1 * window, n=n))



 pwr0 = (np.abs(X0) ** 2) / (n**2 * win_norm + 1e-30)

 pwr1 = (np.abs(X1) ** 2) / (n**2 * win_norm + 1e-30)
```

```
 pwr0_acc += pwr0

 pwr1_acc += pwr1

 X0_last = X0

 X1_last = X1



 raw_max0_last = raw_max0

 raw_max1_last = raw_max1

 scale_last = s

 clip_code_last = clip_code



 pwr0 = pwr0_acc / avg_frames

 pwr1 = pwr1_acc / avg_frames

 pwr = 0.5 * (pwr0 + pwr1)
```

Appendix 1.3 Peak-bin selection and SNR estimation

```
idxs = np.where(band_mask)[0]

 peak_idx = int(idxs[np.argmax(pwr[idxs])])

 peak_hz = float(freq[peak_idx])

 peak0_db = db10(float(pwr0[peak_idx]))

 peak1_db = db10(float(pwr1[peak_idx]))
```

Appendix 1.4 Coherence, phase calibration application, and AoA gating

```
def aoa_from_phase(delta_phi: float, freq_hz: float, baseline_m: float) ->
float:

 lam = 299_792_458.0 / freq_hz
```

```
 arg = (lam * delta_phi) / (2.0 * math.pi * baseline_m)

 arg = max(-1.0, min(1.0, arg))

 return math.degrees(math.asin(arg))


coh = float("nan")

 phase_diff = float("nan")

 aoa = float("nan")

 if X0_last is not None and X1_last is not None:

 s0 = X0_last[lo:hi]

 s1 = X1_last[lo:hi]

 num = np.sum(s0 * np.conj(s1))

 den = math.sqrt(float(np.sum(np.abs(s0)**2) * np.sum(np.abs(s1)**2)) + 1e-30)

 coh = float(np.abs(num) / (den + 1e-30))

 phase_diff = float(np.angle(num)) - float(phase_cal)

 phase_diff = (phase_diff + math.pi) % (2.0 * math.pi) - math.pi


 if (coh >= min_coh) and (snr_db >= min_snr) and not (clip_frac0 >= clip_warn
or clip_frac1 >= clip_warn):

 aoa = aoa_from_phase(phase_diff, peak_hz, baseline_m)
```

Appendix 1.5 GPS acquisition (serial NMEA) and heading estimation

```
class SerialGPS:

 """Lightweight NMEA reader wrapper (RMC/GGA) using pynmeagps + pyserial."""
```

```
def __init__(self, port: str, baud: int, timeout_s: float = 0.2):
 import serial # type: ignore
 from pynmeagps import NMEAReader # type: ignore
 self._serial = serial.Serial(port, baudrate=baud, timeout=timeout_s)
 self._nmr = NMEAReader(self._serial)
 self.last = {
 "lat": float("nan"),
 "lon": float("nan"),
 "status": None, # RMC status: 'A' valid, 'V' void
 "quality": None, # GGA quality: 0 invalid, 1 GPS, 2 DGPS, ...
 "sats": None, # GGA numSV
 "hdop": None,
 }


 def poll(self, max_msgs: int = 6) -> None:
 # Read a few messages to refresh cached fix. Non-blocking due to serial
timeout.
 for _ in range(max_msgs):
 try:
 _raw, msg = self._nmr.read()
 except Exception:
 break
 if not msg:
```

```
 continue

 msg_id = getattr(msg, "msgID", "")

 if msg_id == "RMC":

 status = getattr(msg, "status", None)

 lat = getattr(msg, "lat", None)

 lon = getattr(msg, "lon", None)

 if lat is not None and lon is not None:

 try:

 self.last["lat"] = float(lat)

 self.last["lon"] = float(lon)

 except Exception:

 pass

 self.last["status"] = status

 elif msg_id == "GGA":

 self.last["sats"] = getattr(msg, "numSV", None)

 self.last["quality"] = getattr(msg, "quality", None)

 # Library differs by version; try common attribute names

 self.last["hdop"] = getattr(msg, "HDOP", None) if hasattr(msg, "HDOP") else
getattr(msg, "hdop", None)



 def close(self) -> None:

 try:

 self._serial.close()
```

```
except Exception:
 pass


def read_gps_latlon(gps_kind, gps_handle):
 """Return lat, lon, rmc_status, gga_quality, gga_sats, hdop."""
 if gps_kind == "gpsd" and gps_handle is not None:
 try:
 rep = gps_handle.next()
 if isinstance(rep, dict) and rep.get("class") == "TPV":
 lat = float(rep.get("lat", float("nan")))
 lon = float(rep.get("lon", float("nan")))
 return lat, lon, None, None, None, None
 except Exception:
 pass
 return float("nan"), float("nan"), None, None, None, None


 if gps_kind == "serial" and gps_handle is not None:
 try:
 gps_handle.poll()
 lat = float(gps_handle.last.get("lat", float("nan")))
 lon = float(gps_handle.last.get("lon", float("nan")))
 status = gps_handle.last.get("status", None)
```

```
qual = gps_handle.last.get("quality", None)
 sats = gps_handle.last.get("sats", None)
 hdop = gps_handle.last.get("hdop", None)


 try:
 qual = int(qual) if qual is not None else None
 except Exception:
 qual = None
 try:
 sats = int(sats) if sats is not None else None
 except Exception:
 sats = None
 try:
 hdop = float(hdop) if hdop is not None else None
 except Exception:
 hdop = None


 return lat, lon, status, qual, sats, hdop
 except Exception:
 return float("nan"), float("nan"), None, None, None, None


 return float("nan"), float("nan"), None, None, None, None
```

```
heading_deg = float("nan")

 if np.isfinite(lat) and np.isfinite(lon) and np.isfinite(prev_lat) and
np.isfinite(prev_lon) and prev_t is not None:

 phi1 = math.radians(prev_lat); phi2 = math.radians(lat)

 dlam = math.radians(lon - prev_lon)

 y = math.sin(dlam) * math.cos(phi2)

 x = math.cos(phi1) * math.sin(phi2) - math.sin(phi1) * math.cos(phi2) *
math.cos(dlam)

 heading_deg = (math.degrees(math.atan2(y, x)) + 360.0) % 360.0
```

Appendix 1.6 CSV schema and row logging

```
headers = [

 "sys_time_unix", "sys_time_iso",

 "gps_lat", "gps_lon", "gps_status", "gps_quality", "gps_sats", "gps_hdop",
"heading_deg",

 "freq_hz", "sample_rate_hz", "bandwidth_hz", "fft_n",

 "gain_rx0", "gain_rx1", "bias_tee",

 "band_dBFS_rx0", "band_dBFS_rx1",

 "peak_hz", "peak_dBFS_rx0", "peak_dBFS_rx1",

 "noise_dBFS", "excess_dBFS", "activity_frac",

 "RSSI_pct_rx0", "RSSI_pct_rx1",

 "snr_db", "coh", "phase_diff_rad", "aoa_deg", "aoa_ok",

 "clip_frac_rx0", "clip_frac_rx1",

 "raw_max0", "raw_max1", "scale", "clip_code",
```

```
"baseline_m"
 ]


 wr.writerow([
 f"{now:.3f}", iso,
 f"{lat:.8f}" if np.isfinite(lat) else "",
 f"{lon:.8f}" if np.isfinite(lon) else "",
 gps_status if gps_status is not None else "",
 gps_quality if gps_quality is not None else "",
 gps_sats if gps_sats is not None else "",
 f"{gps_hdop:.2f}" if (gps_hdop is not None) else "",
 f"{heading_deg:.3f}" if np.isfinite(heading_deg) else "",
 f"{tune_hz:.0f}", f"{sr:.0f}", f"{bw:.0f}", n,
 gain0, gain1, int(bias_tee),
 f"{band0_db:.6f}", f"{band1_db:.6f}",
 f"{peak_hz:.0f}", f"{peak0_db:.6f}", f"{peak1_db:.6f}",
 f"{noise_db:.6f}", f"{excess_db:.6f}", f"{activity_frac:.6f}",
 f"{rssi0:.3f}", f"{rssi1:.3f}",
 f"{snr_db:.6f}",
 f"{coh:.8f}" if np.isfinite(coh) else "",
 f"{phase_diff:.8f}" if np.isfinite(phase_diff) else "",
 f"{aoa:.6f}" if np.isfinite(aoa) else "",
```

```
aoa_ok,

f"{clip_frac0:.8f}", f"{clip_frac1:.8f}",

raw_max0_last, raw_max1_last, f"{scale_last:g}", clip_code_last,

f"{baseline_m:.6f}",

])

f.flush()

if np.isfinite(lat) and np.isfinite(lon):

prev_lat, prev_lon = lat, lon

prev_t = now
```

Appendix 2.1 Metric selection (peak/band/rssi) channel combination

```
def _available_metric(df: pd.DataFrame, metric: str) -> str:
 metric = metric.lower()
 if metric in ("auto", "best"):
 # Prefer peak > band > rssi
 if {"peak_dBFS_rx0","peak_dBFS_rx1"}.issubset(df.columns):
 return "peak"
 if {"band_dBFS_rx0","band_dBFS_rx1"}.issubset(df.columns):
 return "band"
 if {"RSSI_pct_rx0","RSSI_pct_rx1"}.issubset(df.columns):
 return "rssi"
 return "unknown"
 return metric


def choose_raw_metric(df: pd.DataFrame, metric: str, channel: str) ->
Tuple[np.ndarray, str, bool]:
 """
 Returns: (raw_values, label, is_dbfs)
 - raw_values is either dBFS (for peak/band) or fraction 0..1 (for rssi)
 """
 metric = _available_metric(df, metric)
 channel = channel.lower()
```

```
if metric == "rssi":
 if not {"RSSI_pct_rx0","RSSI_pct_rx1"}.issubset(df.columns):
 raise SystemExit("RSSI columns not found (expected RSSI_pct_rx0,RSSI_pct_rx1).")
 r0 = pd.to_numeric(df["RSSI_pct_rx0"], errors="coerce").to_numpy() / 100.0
 r1 = pd.to_numeric(df["RSSI_pct_rx1"], errors="coerce").to_numpy() / 100.0
 if channel == "rx0":
 return r0, "RSSI rx0", False
 if channel == "rx1":
 return r1, "RSSI rx1", False
 if channel == "mean":
 return 0.5*(r0+r1), "RSSI mean(rx0,rx1)", False
 return np.maximum(r0, r1), "RSSI max(rx0,rx1)", False

 if metric in ("peak", "band"):
 c0 = f"{metric}_dBFS_rx0"
 c1 = f"{metric}_dBFS_rx1"
 if not {c0,c1}.issubset(df.columns):
 raise SystemExit(f"{metric.upper()} columns not found (expected {c0},{c1}).")
 v0 = pd.to_numeric(df[c0], errors="coerce").to_numpy()
 v1 = pd.to_numeric(df[c1], errors="coerce").to_numpy()
 if channel == "rx0":
 return v0, f"{metric.upper()} dBFS rx0", True
```

```
if channel == "rx1":

 return v1, f"{metric.upper()} dBFS rx1", True

 if channel == "mean":

 return 0.5*(v0+v1), f"{metric.upper()} mean dBFS", True

 return np.maximum(v0, v1), f"{metric.upper()} max dBFS", True



 raise SystemExit("Did not find columns I recognize. Need either RSSI_pct_rx*,
peak_dBFS_rx*, or band_dBFS_rx*.")
```

Appendix 2.2 True-bearing derivation (phase + baseline + frequency + heading)

```
def compute_bearing_from_phase(df: pd.DataFrame, phi_col: str, baseline_m:
Optional[float], freq_hz: Optional[float], heading_col: Optional[str]):

 if phi_col not in df.columns:

 return df

 if freq_hz is None:

 if 'freq_hz' in df.columns:

 lam = 299792458.0 / pd.to_numeric(df['freq_hz'], errors='coerce')

 else:

 print('[warn] No frequency provided; cannot compute bearing_deg from phase.');

 return df

 else:

 lam = 299792458.0 / float(freq_hz)
```

```
if baseline_m is None:
 if 'baseline_m' in df.columns:
 baseline = pd.to_numeric(df['baseline_m'], errors='coerce')
 else:
 print('[warn] No baseline_m; cannot compute bearing_deg from phase.');
 return df
 else:
 baseline = float(baseline_m)


 phi = pd.to_numeric(df[phi_col], errors='coerce').to_numpy()
 phi = np.arctan2(np.sin(phi), np.cos(phi)) # wrap to [-pi,pi]
 if np.isscalar(lam):
 arg = (lam * phi) / (2.0*np.pi*baseline)
 else:
 lamv = lam.to_numpy()
 arg = (lamv * phi) / (2.0*np.pi*baseline)
 arg = np.clip(arg, -1.0, 1.0)
 rel_deg = np.degrees(np.arcsin(arg))


 if heading_col and heading_col in df.columns:
 hdg = pd.to_numeric(df[heading_col], errors='coerce').to_numpy()
 bear = (hdg + rel_deg) % 360.0
```

```
else:
 bear = (rel_deg + 360.0) % 360.0


 out = df.copy()
 out['bearing_deg'] = bear
 return out
```

Appendix 2.3 Hover-safe grid aggregation

```
def grid_aggregate(df: pd.DataFrame, raw: np.ndarray, grid_m: float, stat: str)
-> pd.DataFrame:
 """Aggregate points into a meter grid; output one row per cell with
representative raw metric."""
 if grid_m <= 0:
 out = df.copy()
 out["_raw"] = raw
 out["_n"] = 1
 return out


 # Local metric CRS (AEQD) for stable meters
 clat = float(df['gps_lat'].mean()); clon = float(df['gps_lon'].mean())
 crs = f"+proj=aeqd +lat_0={clat} +lon_0={clon} +x_0=0 +y_0=0 +units=m
+R=6378137"
 fwd = Transformer.from_crs("EPSG:4326", crs, always_xy=True)
 inv = Transformer.from_crs(crs, "EPSG:4326", always_xy=True)
```

```
lats = df["gps_lat"].to_numpy()

lons = df["gps_lon"].to_numpy()

xy = np.array([fwd.transform(lon, lat) for lon, lat in zip(lons, lats)])

x = xy[:,0]; y = xy[:,1]



cx = np.floor(x / grid_m).astype(int)

cy = np.floor(y / grid_m).astype(int)



tmp = df.copy()

tmp["_raw"] = raw

tmp["_cx"] = cx

tmp["_cy"] = cy



stat = stat.lower()

def agg_raw(s: pd.Series) -> float:

a = pd.to_numeric(s, errors="coerce").to_numpy()

a = a[np.isfinite(a)]

if a.size == 0:

return float("nan")

if stat == "max":

return float(np.max(a))

if stat == "mean":
```

```
 return float(np.mean(a))

 if stat == "median":

 return float(np.median(a))

 if stat == "p90":

 return float(np.percentile(a, 90))

 if stat == "p95":

 return float(np.percentile(a, 95))

 raise SystemExit("grid-stat must be one of: max, mean, median, p90, p95")


 def agg_bearing(s: pd.Series) -> float:

 a = pd.to_numeric(s, errors="coerce").to_numpy()

 return circular_mean_deg(a)


 gb = tmp.groupby(["_cx","_cy"], as_index=False)

 out = gb.agg(

 gps_lat=("gps_lat", "mean"),

 gps_lon=("gps_lon", "mean"),

 _raw=("_raw", agg_raw),

 _n=("_raw", "count"),

 bearing_deg=("bearing_deg", agg_bearing) if "bearing_deg" in tmp.columns else
("gps_lat", "size"),

 )

 # Fix bearing column when not present: groupby created nonsense; drop it
```

```
if "bearing_deg" not in tmp.columns:
 out = out.drop(columns=["bearing_deg"], errors="ignore")

 return out

dfA = grid_aggregate(df, raw, args.grid_m, args.grid_stat)
rawA = pd.to_numeric(dfA["_raw"], errors="coerce").to_numpy()
```

Appendix 2.4 dBFS -> weight scaling and non-linear shaping

```
def normalize01_linear(v, vmin, vmax):
 denom = max(1e-9, (vmax - vmin))
 return np.clip((v - vmin) / denom, 0.0, 1.0)

def apply_scale_shape(w01: np.ndarray, mode: str, gamma: float, logistic_k:
float, logistic_x0: Optional[float], raw_dbfs: Optional[np.ndarray], vmin:
float, vmax: float):
 mode = (mode or "linear").lower()
 w = np.clip(w01, 0.0, 1.0)
 if mode == "linear":
 return w
 if mode == "gamma":
 # gamma > 1 => compress mid/low values, emphasize peaks
 g = max(1e-6, float(gamma))
 return np.power(w, g)
```

```
 if mode == "logistic":
 if raw_dbfs is None:
 # fall back: logistic in normalized space
 x = w
 x0 = 0.5 if logistic_x0 is None else float(logistic_x0)
 else:
 x = raw_dbfs
 x0 = (0.5*(vmin+vmax)) if logistic_x0 is None else float(logistic_x0)
 k = float(logistic_k)
 return 1.0 / (1.0 + np.exp(-k * (x - x0)))
 raise SystemExit(f"Unknown --scale {mode}. Use linear|gamma|logistic.")


vmin = float(args.dbfs_min)
 vmax = float(args.dbfs_max)
 if is_dbfs:
 auto = args.dbfs_auto.lower()
 if auto != "none":
 if auto == "p10p99":
 vmin = percentile(rawA, 10)
 vmax = percentile(rawA, 99)
 elif auto == "p5p95":
 vmin = percentile(rawA, 5)
```

```
 vmax = percentile(rawA, 95)

 elif auto == "p1p99":

 vmin = percentile(rawA, 1)

 vmax = percentile(rawA, 99)

 # sanity

 if not (math.isfinite(vmin) and math.isfinite(vmax)) or (vmax - vmin) < 1.0:

 vmin = float(args.dbfs_min)

 vmax = float(args.dbfs_max)



 w01 = normalize01_linear(rawA, vmin, vmax)

 w = apply_scale_shape(w01, args.scale, args.gamma, args.logistic_k,
args.logistic_x0, rawA, vmin, vmax)

 dfA["_w"] = w

 dfA["_wlabel"] = f"{raw_label} → weight (dbfs {vmin:.1f}..{vmax:.1f},
{args.scale})"

 else:

 # RSSI already 0..1; apply shaping only

 w01 = np.clip(rawA, 0.0, 1.0)

 w = apply_scale_shape(w01, args.scale, args.gamma, args.logistic_k,
args.logistic_x0, None, 0.0, 1.0)

 dfA["_w"] = w

 dfA["_wlabel"] = f"{raw_label} → weight ({args.scale})"



 # Drop low weights (after shaping)
```

```
dfA = dfA[dfA["_w"] >= float(args.min_weight)]

 if len(dfA) == 0:

 raise SystemExit("All points filtered out. Lower --min-weight or adjust scaling.")
```

Appendix 2.5 Map construction and overlays (route + heat layer + markers)

```
# Bounding boxes

 if args.mode == "extent":

 lat_min, lon_min, lat_max, lon_max = bbox_from_extent(df_gps)

 else:

 lat_min, lon_min, lat_max, lon_max = bbox_from_first_last(df_gps)

 plat_min, plon_min, plat_max, plon_max = pad_bbox_wgs84(lat_min, lon_min, lat_max, lon_max, args.padding_m)



 # Map center

 center_lat = 0.5*(plat_min + plat_max)

 center_lon = 0.5*(plon_min + plon_max)



 # Build map

 m = folium.Map(location=[center_lat, center_lon], zoom_start=15, control_scale=True, prefer_canvas=True)

 add_tilelayer(m, args.tiles)



 # Full GPS route polyline (unaggregated) so you always see start → end
```

```
if not args.no_track:

 step = max(1, int(args.track_subsample))

 track_pts= list(zip(df_gps["gps_lat"].iloc[::step].to_numpy().tolist(),

 df_gps["gps_lon"].iloc[::step].to_numpy().tolist()))

 if len(track_pts) >= 2:

 folium.PolyLine(track_pts,                    weight=int(args.track_weight),
opacity=float(args.track_opacity), tooltip="Route").add_to(m)

 if args.mark_start_end:

 folium.CircleMarker(location=track_pts[0],    radius=5,    tooltip="START",
fill=True).add_to(m)

 folium.CircleMarker(location=track_pts[-1],    radius=5,    tooltip="END",
fill=True).add_to(m)



 rectangle(m, lat_min, lon_min, lat_max, lon_max, weight=2, opacity=0.9)

 rectangle(m, plat_min, plon_min, plat_max, plon_max, weight=1, opacity=0.6)



 # Heat points

 heat_data = [[float(r["gps_lat"]), float(r["gps_lon"]), float(r["_w"])] for
_, r in dfA.iterrows()]

 HeatMap(

 data=heat_data,

 radius=args.radius,

 blur=args.blur,

 min_opacity=0.15,
```

```
max_zoom=18,
 name=str(dfA["_wlabel"].iloc[0]) if "_wlabel" in dfA.columns else "Heat",
 ).add_to(m)

 # Top points markers (after aggregation)
 top = dfA.sort_values("_w", ascending=False).head(10)
 for _, r in top.iterrows():
 rawv = float(r["_raw"]) if math.isfinite(float(r["_raw"])) else float("nan")
 tip = f"{raw_label}: {rawv:.1f}{' dBFS' if is_dbfs else ''} |
w={float(r['_w']):.2f} | n={int(r.get('_n',1))}"
 folium.CircleMarker(
 location=(float(r["gps_lat"]), float(r["gps_lon"])),
 radius=3,
 opacity=0.95,
 tooltip=tip,
 fill=True
 ).add_to(m)

 # AoA rays if requested
 if args.aoa_maxlen_m > 0:
 add_aoa_rays_if_present(m, dfA, args.aoa_maxlen_m)

 folium.LayerControl(collapsed=False).add_to(m)
```

```
m.fit_bounds([[plat_min, plon_min], [plat_max, plon_max]])


 args.out.parent.mkdir(parents=True, exist_ok=True)

 m.save(str(args.out))
```

Appendix 2.6 AoA ray overlay

```
def add_aoa_rays_if_present(m: folium.Map, df: pd.DataFrame, maxlen_m: float =
500.0):

 if "bearing_deg" not in df.columns:

 return 0

 geod = Geod(ellps="WGS84")

 count = 0

 for _, row in df.iterrows():

 lat = row.get("gps_lat"); lon = row.get("gps_lon"); brg =
row.get("bearing_deg")

 if pd.isna(lat) or pd.isna(lon) or pd.isna(brg):

 continue

 lon2, lat2, _ = geod.fwd(float(lon), float(lat), float(brg), float(maxlen_m))

 folium.PolyLine([(lat, lon), (lat2, lon2)], weight=2, opacity=0.55,
tooltip=f"AoA {brg:.1f}°").add_to(m)

 count += 1

 return count
```

MAP-MAKING PARAMETERS AND DEFAULTS **APPENDIX 3/1**

Group	Parameter (CLI)	Default	Unit	What it does (short)
I/O	csv (positional)	-	-	Input CSV produced by logger; must include gps_lat and gps_lon.
I/O	--out	heatmap.html	path	Output HTML path for the map.
Tiles	--tiles	osm	preset/URL	Basemap tile preset or URL template.
Extent & framing	--mode	first-last	choice	Map framing: inner rectangle from first→last fixes, or full extent.
Extent & framing	--padding-m	100	m	Extra margin around the inner rectangle (meters).
Heat layer	--radius	18	px	Heat-point radius.
Heat layer	--blur	25	px	Heat blur width.
Weighting & filtering	--min-weight	0	0-1	Drop points below this normalized weight.
Track overlay	--no-track	False	flag	Disable drawing the full GPS route polyline.

Track overlay	--track-subsample	1	-	Draw every Nth GPS point for the route polyline.
Track overlay	--track-weight	3	px	Route polyline width.
Track overlay	--track-opacity	0,85	0–1	Route polyline opacity.
Track overlay	--mark-start-end	False	flag	Mark start and end points on the route.
Metric selection	--metric	auto	choice	Metric to visualize: auto/peak/band/rssi. Auto prefers peak→band→rssi based on available columns.
Channel combine	--channel	max	choice	Combine rx0/rx1 into one: rx0/rx1/max/mean.
dBFS scaling	--dbfs-min	-95	dBFS	Lower bound for scaling dBFS → weight.
dBFS scaling	--dbfs-max	-25	dBFS	Upper bound for scaling dBFS → weight.
dBFS scaling	--dbfs-auto	none	choice	Auto-pick dbfs-min/max from run percentiles: none/p10p99/p5p95/p1p99.

Weight shaping	--scale	linear	choice	Optional non-linear shaping: linear/gamma/logistic.
Weight shaping	--gamma	2,5	-	Gamma exponent when --scale gamma (>1 emphasizes peaks).
Weight shaping	--logistic-k	0,7	-	Logistic slope when --scale logistic.
Weight shaping	--logistic-x0	None	dBFS	Logistic center point; default is midpoint of dbfs-min/max.
Hover-safe aggregation	--grid-m	0	m	If >0, aggregate samples into meter grid cells to reduce hover/stop overweighting.
Hover-safe aggregation	--grid-stat	p90	choice	Statistic per cell: max/mean/median/p90/p95.
AoA overlay	--aoa-maxlen-m	0	m	If >0 and bearing_deg exists, draw AoA rays of this length.
Bearing compute	--compute-bearing	False	flag	Derive bearing_deg from phase_diff_rad + baseline + freq + optional heading.

Bearing compute	--heading-col	platform_yaw_deg	col name	Heading column (degrees) used to rotate relative AoA into true bearing.
Bearing compute	--phi-col	phase_diff_rad	col name	CSV column for Δφ (radians).
Bearing compute	--baseline-m	None	m	Override baseline length if not present in CSV as baseline_m.
Bearing compute	--freq-hz	None	Hz	Override frequency if not present in CSV as freq_hz.
dBFS scaling	--dbfs-auto	none	choice	Auto-pick dbfs-min/max from run percentiles: none/p10p99/p5p95/p1p99.

UML MICRO WORKFLOW FOR MAP-MAKING (CSV TO HTML) APPENDIX 4