

Android GNSS Measurements under Spoofing and Interference

Andrea Botticella^{*†}
andrea.botticella@studenti.polito.it

Renato Mignone^{*†}
renato.mignone@studenti.polito.it

Elia Innocenti^{*†}
elia.innocenti@studenti.polito.it

Simone Romano^{*†}
simone.romano2@studenti.polito.it

ABSTRACT

This study explores the behavior of GNSS navigation on a commercial Android smartphone by analyzing raw measurements collected in both static and dynamic conditions. Using Google’s GNSS Logger and MATLAB-based post-processing, we evaluate PVT (Position, Velocity and Time), signal quality, and clock stability under normal operation, under spoofed inputs (false positions/delays), and interference. The results highlight vulnerabilities to software spoofing—positional errors with minimal signal-quality degradation—and tracking instability caused by interference.

1 INTRODUCTION

Global Navigation Satellite Systems (GNSS) are fundamental to modern positioning services, widely embedded in smartphones and critical infrastructure. Though widely adopted, GNSS signals are inherently vulnerable to **spoofing**—an attack technique where counterfeit signals deceive the receiver into computing false location or timing information. To develop robust mitigation strategies, it is essential to understand how smartphone GNSS systems behave under both normal and manipulated conditions.

This work investigates the GNSS measurement behavior of a consumer-grade **Android smartphone** across different operating scenarios. We analyze baseline performance by collecting and processing raw GNSS data in two real-world environments: a static session on viewpoint of *Monte dei Cappuccini* and a dynamic session aboard a tram in urban *Turin*. In addition to this baseline study, we simulate spoofing by injecting false position inputs into the processing pipeline and, separately, by introducing artificial delays, as well as evaluating signal degradation from electromagnetic interference. The remainder of this report is organized as follows.

- **Section 2** describes the experimental setup, including device configuration, data collection procedures, and the processing pipeline.
- **Section 3** presents results and discussion, contrasting static versus dynamic performance, examining spoofed-location impacts, analyzing delay effects, and evaluating an interference scenario.
- **Section 4** summarizes the key findings and outlines directions for future work.

2 METHODS

2.1 Devices and Software

We used a Samsung Galaxy A51 with Android 13 for this experiment. GNSS Logger v3.1.0.4 [8] was chosen due to its unrestricted

access to raw GNSS measurements, compatibility with newer Android APIs, and ability to record detailed GNSS data that is suitable for precise analysis. MATLAB R2024b [10] was employed to handle data because it comes with Google’s GNSS toolbox [7], which accommodates robust analysis and visualization of GNSS measurements and position solutions.

2.2 Data Collection Procedure

Two distinct 5-minute GNSS data logging sessions were conducted on 3 May 2025, under cloudy weather conditions, using the GNSS Logger app configured with the following settings enabled:

- **GNSS Location:** to capture location data.
- **GNSS Measurements:** to log raw GNSS measurements.
- **Navigation Messages:** to capture navigation data.
- **GnssStatus:** to log GNSS status information.
- **Sensors:** to capture sensor data.

The sessions were designed to capture both static and dynamic GNSS performance, with the following details:

- Static Scenario:** performed on the viewpoint of *Monte dei Cappuccini, Turin*, starting at 10:35:20. The device was stationary throughout the entire session, providing baseline measurements.
- Dynamic Scenario:** conducted on tram line 15 from *Piazza Castello* to *Piazza Vittorio Veneto*, starting at 10:00:21, simulating a typical urban mobility scenario.

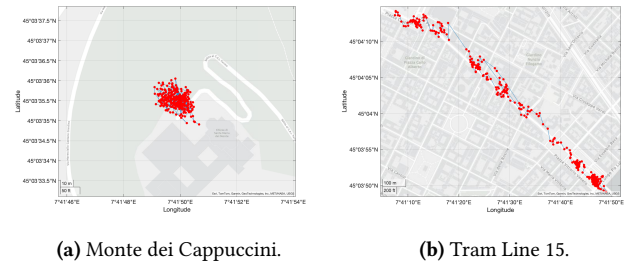


Figure 1: Comparison of GNSS data: static (a) and dynamic (b) scenarios.

2.3 Processing Pipeline

The raw GNSS data from the GNSS Logger served as the input dataset for MATLAB. Processing involved a scripted workflow via `ProcessGnssMeasScript.m` [12], where the following steps were executed:

1. **Filtering:** data points not meeting predefined quality thresholds, such as signal strength or satellite geometry, were excluded to improve accuracy.

^{*}The authors collaborated closely in developing this project.

[†]All the authors are students at Politecnico di Torino, Turin, Italy.

2. **Measurement Extraction:** pseudorange and Doppler measurements were computed from GNSS timestamps and satellite transmission data.
3. **Weighted Least Squares (WLS) Positioning:** applied to derive precise positioning and clock bias estimates.
4. **Visualization and Comparison:** output plots from MATLAB, including pseudorange, pseudorange rates, and position solutions, were generated to facilitate comparative analysis of the static and dynamic scenarios.

2.4 Spoofed-Input Configuration

Spoofing scenarios were emulated by introducing artificial variations to the recorded GNSS data through MATLAB processing. Specifically, mock positions were assigned by adjusting the parameter `spooof.position`, which represents modified latitude, longitude, and altitude coordinates. Additionally, artificial time delays were tested by adjusting the `spooof.delay` parameter, typically in milliseconds, to mimic delayed GNSS signal arrival, and `cfg.t_start` parameter to make the spoofed signal appear some seconds after the start of the session. Such configurations facilitated evaluation of the impact of spoofing scenarios on position estimation reliability and accuracy.

2.5 Interference Scenario

The experimental setup involved enclosing a smartphone within **three layers of aluminum foil** while simultaneously exposing it to nearby **sources of electromagnetic interference**, including two active smartphones engaged in a call and a Bluetooth-enabled device. This configuration created a non-ideal environment for GNSS signal reception. The aluminum enclosure acted as a rudimentary **Faraday cage**, partially obstructing direct **line-of-sight** between the device's antenna and satellites, weakening incoming signals. Concurrently, the proximity of active communication devices introduced broadband RF interference, likely overlapping with GNSS frequency bands and degrading signal quality further.

3 RESULTS AND DISCUSSIONS

3.1 Baseline Performance: Static vs. Dynamic

Comparing the static and dynamic scenarios highlights key differences in GNSS signal behavior, measurement stability, and receiver performance. Although both datasets were collected under similar atmospheric conditions, the receiver's motion in the dynamic case introduced visible changes across all GNSS indicators.

Pseudoranges vs Time.

The pseudorange measurements reveal distinct trends across scenarios. In Fig. 2a **more satellites** (e.g., SVs 27, 4, 20, 5, 16, 28, 29, 18) are consistently tracked, exhibiting **flat trajectories** with minimal variation (range: $\sim 2.1\text{--}2.5 \times 10^7$ m). This suggests stable signal acquisition under favorable conditions. In contrast, Fig. 2b displays **fewer visible satellites** (e.g., SVs 28, 29, 31, 12, 18), with several signals showing abrupt shifts or non-linear behavior (range: $\sim 2.0\text{--}2.6 \times 10^7$ m). The reduced satellite count in Plot B implies **challenging signal reception**, possibly due to environmental interference or motion-induced signal degradation.

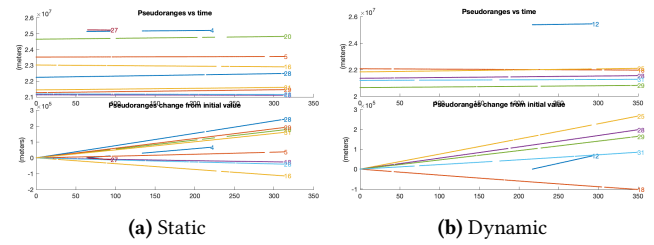


Figure 2: Comparison of Pseudoranges vs Time

Pseudorange Change from Initial Value.

Deviations from initial values further highlight differences. In Fig. 2a, all tracked satellites follow **shallow linear trends** ($\pm 0.5\text{--}2$ m), reflecting predictable signal dynamics. The consistency across satellites (e.g., SVs 27, 28, 29) suggests robust tracking of both strong and weaker signals, including those from higher-elevation satellites. Conversely, Fig. 2b shows **steeper slopes** (up to ± 3 m) and fragmented trends, with notable examples like SV 12 (sudden jump after ~ 200 s) and SV 18 (continuous negative drift). The variability in Plot B indicates **intermittent signal loss or multipath effects**, particularly for satellites with marginal signal strength.

Positioning and Speed.

In the plots depict positional estimates. Fig. 3a displays a clustered distribution around its median ($45.059862^\circ, 7.697191^\circ$), with a 50% confidence ellipse spanning $\sim \pm 15$ m. This indicates stable positioning with minimal drift. In contrast, Fig. 3b shows a widely scattered pattern centered at ($45.066832^\circ, 7.692336^\circ$), featuring a 50% ellipse extending $\sim \pm 400$ m. The median in Plot B lacks practical relevance due to extreme positional variability, unlike Plot A's median, which accurately represents stationary behavior.

The middle plots highlight horizontal speed trends. Fig. 3a exhibits near-zero speeds (< 2 m/s) with minor oscillations, consistent with negligible movement. Conversely, Fig. 3b reveals erratic fluctuations, including a peak exceeding 20 m/s.

The bottom plots compare HDOP and satellite visibility. Fig. 3a maintains low HDOP (< 5) with consistent visibility of 8-10 satellites, reflecting robust signal geometry. In contrast, Fig. 3b shows elevated HDOP (up to ~ 10) alongside reduced satellite counts (4-6), indicating degraded tracking conditions.

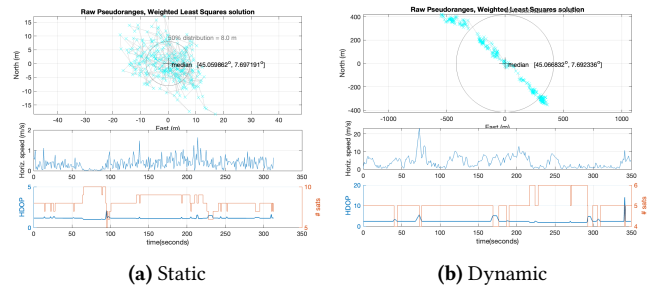


Figure 3: Comparison of Position, Speed, and HDOP

State Offsets and Timing Bias.

The **top plots** illustrate position state offsets relative to the median. In Fig. 4a, deviations in latitude, longitude, and altitude remain tightly constrained within $\sim \pm 50$ m, with smooth, gradual fluctuations indicating stable signal tracking. In contrast, Fig. 4b exhibits

extreme offsets exceeding ± 500 m, punctuated by abrupt spikes (e.g., a ~ 600 m jump near 150 seconds). The **second plot** shows clock bias (common bias) trends. Both plots exhibit linear growth, but in Fig. 4a increases steadily (~ 200 ns over 350 seconds), reflecting predictable receiver clock drift. Fig. 4b escalates more sharply (~ 250 ns over the same interval), with a notable dip aligning with the positional spike in the top plot. This correlation implies transient disruptions affecting both position and timing stability in Plot B. Velocity state estimates (**third plot**) further differentiate the two scenarios. Fig. 4a shows near-zero velocities (all components ≤ 2 m/s), consistent with negligible movement. Fig. 4b, however, displays pronounced oscillations, particularly in the northward direction (peaks ± 20 m/s), indicating rapid or inconsistent motion. The **bottom plots** depict common frequency offset trends. Fig. 4a maintains a stable trajectory near 0.8 ppm, with minor perturbations. Fig. 4b shows greater variability, including a sharp downward deviation during the positional spike. This suggests that signal degradation or motion-induced errors propagate to both position and frequency estimates in Plot B, unlike the consistent behavior seen in Plot A.

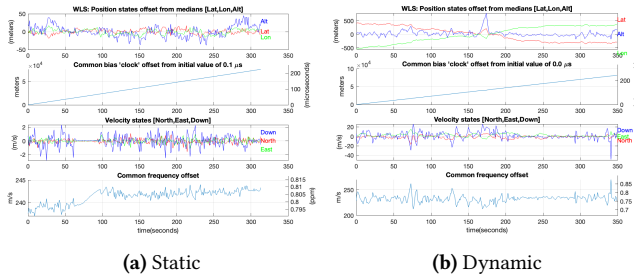


Figure 4: Comparison of WLS states and bias

These measurements highlight the importance of scenario-aware processing: identical hardware, firmware, and atmospheric conditions can yield centimetre-per-second stability on a quiet terrace, yet the very same setup will produce hundred-metre excursions once placed on a moving tram through an urban canyon.

3.2 Impact of Spoofed Position

In a spoofing attack, counterfeit signals are broadcast to mimic GNSS transmissions, often altering their timing, amplitude, or content. Typically, these deceptive signals are sent at higher power than the originals to mislead navigation systems. In our experiment, however, spoofing was implemented purely in software by adjusting the perceived reception time inside the professor’s script to simulate spoofing behavior.

For the spoofed test we manually injected the coordinates of **Piazza Vittorio Veneto** (45.064749, 7.695466) — a location adjacent to our true survey point — directly into the processing script. As shown in the plots, this causes the solver to converge exactly at the Piazza Vittorio Veneto coordinates rather than the true measurement site, introducing a horizontal displacement of several hundred metres. Despite this spatial shift, all raw observables remain essentially unchanged relative to the normal run:

- **Carrier-to-noise density (C/N):** The plot of the signal quality remains unchanged compared to the base case, this because the

signals are just replayed with a different timing to the victim. **Pseudorange residuals:** The overall spread remains the same, but the entire residual distribution is offset by the constant delay corresponding to the spoofed displacement.

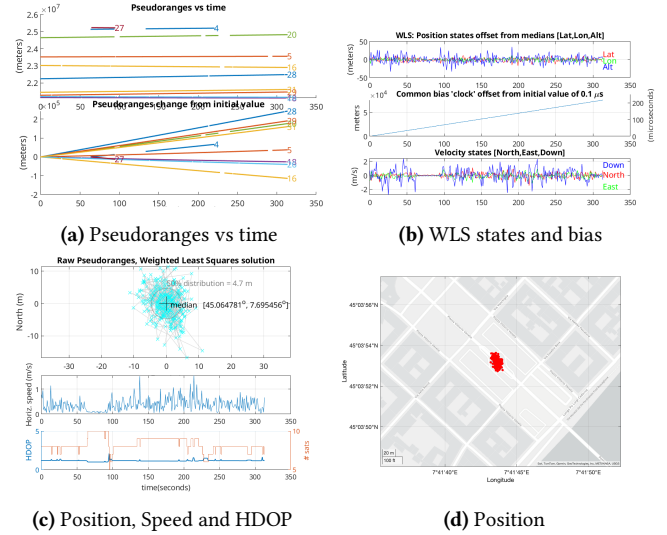


Figure 5: Spoofing scenario plots

Additionally, the 50% (interquartile) range in the median-error distribution shrinks from approximately 8 m in the genuine case to about 4.7 m under spoofing. This reduction occurs because the solver consistently “locks” onto the spoofed coordinate, decreasing variability around that false point.

3.3 Effects of Timing Delays

In this delayed-spoofing scenario we again target the same true survey point but now introduce a software-only replay delay: the spoofer “listens” to genuine signals, waits 1 ms, then injects the spoofed Piazza Vittorio Veneto coordinates starting at 50 s into the run. Key parameters: `cfg.delay = 1e-3`; `cfg.t_start = 50`. As visible in the plots, all observables remain nominal until $t=50$ s, at which point the solver’s estimated position and receiver clock-bias exhibit a clear discontinuity as the spoof takes effect.

- **Position solution:** Prior to 50 s, the estimated coordinate coincides with the true static point. Immediately after 50 s, the solution jumps to the spoofed Piazza Vittorio Veneto location, replicating the static-spoof offset of several hundred metres.
- **Receiver clock-bias:** A sudden step of appears in the estimated receiver clock-bias track, directly corresponding to the 1 ms replay delay needed to shift the range solution by the planar offset.
- **Carrier-to-noise density (C/N):** The C/N time-series shows no amplitude change at $t=50$ s—signal strength is unaffected by delay.
- **Pseudorange residuals:** Residuals maintain the same spread, but their mean shifts abruptly at 50 s by the additional delay δt .

Even a 1 ms replay delay can therefore introduce a large positional error and matching clock-bias shift without altering any standard

224 observables. Detecting such covert delay-based spoofing thus re-
 225 quires monitoring for timing discontinuities.

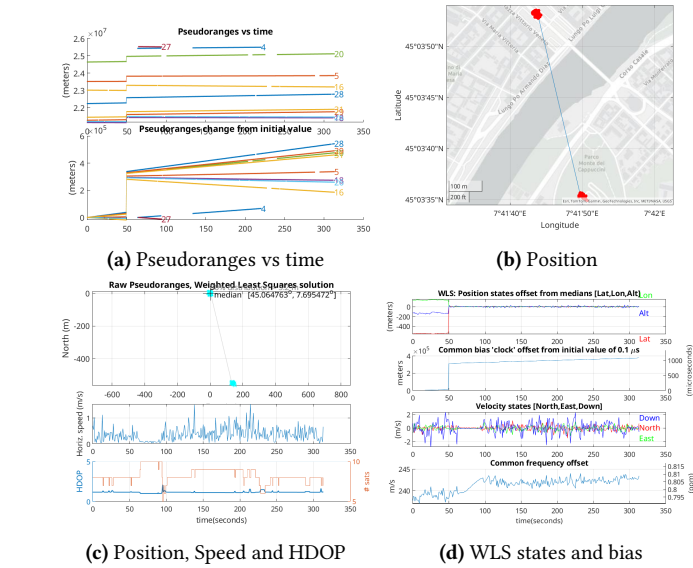


Figure 6: Spoofing plus Delay scenario plots

3.4 Interference Effects

226 Analyzing the logged data reveals several effects of this compro-
 227 mised environment. A key detail is that data collection began before
 228 the foil shielding was applied. During this initial phase, the receiver
 229 operated under normal conditions and successfully acquired a large
 230 number of satellites with strong signals. However, once the shielding
 231 was added, the number of tracked satellites dropped significantly,
 232 and signal quality degraded—clear in the plots by a drop in
 233 C/N values and clock discontinuities.

235 **Pseudorange measurements** (Figure 7a), which estimate the dis-
 236 tance between the receiver and satellites, exhibit gradual drift over
 237 time, indicative of accumulating errors due to signal instability.
 238 The **carrier-to-noise density** (C/N_0) metric (Figure 7b) shows
 239 sharp initial drops followed by slow recovery, reflecting the re-
 240 ceiver's struggle to reacquire and track satellite signals through
 241 the combined effects of shielding and interference. **Velocity es-**
 242 **timates** derived from Doppler shifts display erratic fluctuations,
 243 particularly during periods of heightened interference, suggesting
 244 corrupted phase measurements. **Position solutions** (Figure 7d)
 245 exhibit significant horizontal and vertical deviations from expected
 246 values, with error distributions expanding over time as the receiver
 247 compensates for inconsistent pseudorange inputs.

248 In such scenario, the interplay between physical obstructions and
 249 electromagnetic noise creates a hostile operating environment for
 250 GNSS receivers. The aluminum enclosure not only attenuates sig-
 251 nal power but also **shields them**. Nearby devices, meanwhile,
 252 introduce spurious signals that desensitize the receiver's front-end,
 253 increasing the likelihood of cycle slips (loss of phase lock) and pseu-
 254 dorange outliers. These findings reinforce the sensitivity of GNSS
 255 systems to real-world complexities, where idealized assumptions of
 256 unobstructed, interference-free signal propagation rarely hold. Prac-
 257 tical applications in urban canyons, dense foliage, or near industrial

RF emitters may thus encounter similar challenges, necessitating 258
 robust mitigation strategies. 259

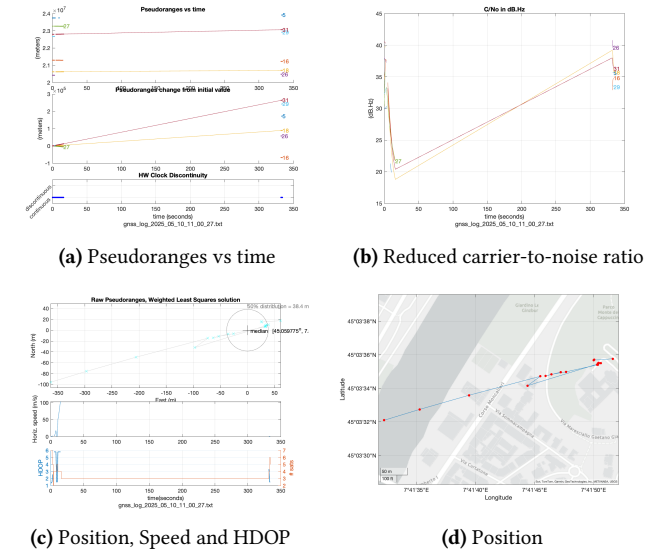


Figure 7: Interference effects on GNSS measurements

4 CONCLUSIONS

260 This research compared GNSS performance on a commercial smart-
 261 phone under static, dynamic, and spoofed scenarios based on raw
 262 measurements and post-processing in MATLAB. Under the static
 263 scenario, the device had consistent position estimates, low clock
 264 drift, and negligible variation in pseudorange and speed. The dy-
 265 namic scenario, on the other hand, demonstrated higher noise in
 266 position, pseudorange rates, and clock bias—predictable effects of
 267 motion and satellite geometry variations. To further assess receiver
 268 robustness, we introduced an interference scenario by shielding
 269 the device with aluminum foil and placing it near two actively
 270 transmitting smartphones. This setup caused a noticeable drop in
 271 signal quality, a reduction in the number of tracked satellites, and
 272 significant degradation in position accuracy. Despite the device
 273 remaining stationary, the solution drifted tens of meters, revealing
 274 the sensitivity of smartphone GNSS to even passive signal obstruc-
 275 tion and nearby RF noise. The spoofing tests indicated that even
 276 basic, software-level injection in the form of taking on false coor-
 277 dinates by the GNSS solver, as it settles on a consistent but incorrect
 278 position with little or no impact on first-order signal measurements
 279 like carrier-to-noise ratio or HDOP. This highlights a significant
 280 vulnerability: smartphone GNSS receivers can be fooled without
 281 revealing obvious degradations in signal quality.

A APPENDIX

A.1 GNSS Logger Screenshots

The following figures show screenshots from the GNSS Logger app, illustrating various features and data visualizations available during the GNSS data collection process of the *Monte dei Cappuccini* test.

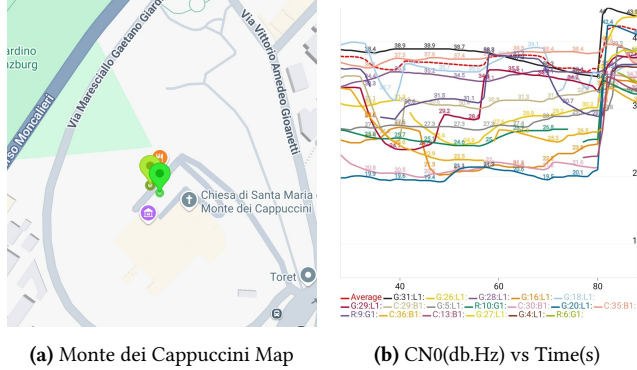


Figure 8: GNSS Logger Map (a) and Plots (b).

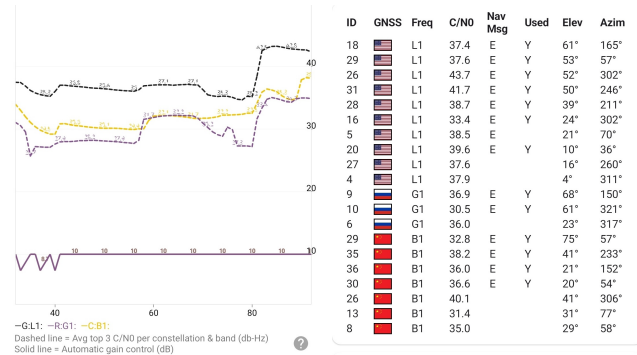


Figure 9: GNSS Logger Spoof/Jam (a) and Status (b).

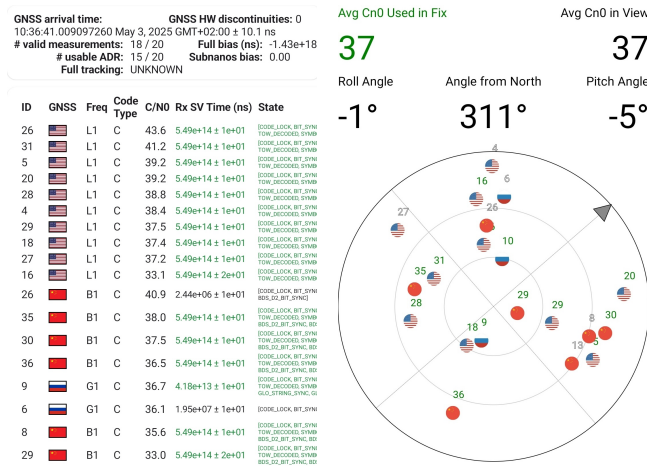


Figure 10: GNSS Logger Measurements (a) and Skyplot (b).

REFERENCES

- [1] 2025. GNSS Planning. Web application. (2025). <https://www.gnssplanning.com/#/settings>
- [2] 2025. LatLongData. Web application. (2025). <https://www.latlongdata.com/elevation/>
- [3] Android Developers. 2025. GnsMeasurement. Android SDK reference. (2025). <https://developer.android.com/reference/android/location/GnsMeasurement>
- [4] Simon Banville and Frank van Diggelen. 2016. Precise GNSS for Everyone: Precise Positioning Using Raw GPS Measurements from Android Smartphones. *GPS World* 27, 11 (November 2016). <http://gpsworld.com/innovation-precise-positioning-using-raw-gps-measurements-from-android-smartphones/>
- [5] Alan Cameron. 2016. Google opens up GNSS pseudoranges. *GPS World* (7 June 2016). <https://www.gpsworld.com/google-opens-up-gnss-pseudoranges/>
- [6] European GNSS Agency (GSA). 2018. *Using GNSS Raw Measurements on Android Devices: Towards better location performance in mass market applications*. Technical Report. GSA GNSS Raw Measurements Task Force. <https://galileognss.eu/wp-content/uploads/2018/05/Using-GNSS-Raw-Measurements-on-Android-devices.pdf>
- [7] Google LLC. 2025. GNSS Analysis Tools v2.0.0.1. GitHub release. (2025). <https://github.com/google/gps-measurement-tools/releases/tag/2.0.0.1>
- [8] Google LLC. 2025. GNSS Logger v3.1.0.4. Android app on Google Play. (2025). <https://play.google.com/store/apps/details?id=com.google.android.apps.gnsslogger>
- [9] Google LLC. 2025. Google Maps. Web application. (2025). <https://www.google.com/maps>
- [10] The MathWorks, Inc. 2024. *MATLAB R2024b*. Natick, Massachusetts, United States. <https://www.mathworks.com/products/matlab.html>
- [11] Frank van Diggelen. 2018. GNSS Measurements Update. In *GSA Raw Measurements Workshop*. Prague, Czech Republic. https://www.gsa.europa.eu/sites/default/files/expo/frank_van_diggelen_keynote_android_gnss_measurements_update.pdf
- [12] WDCSecure. 2025. LabGNSS: material and documentation for the GNSS Lab. GitHub repository. (2025). <https://github.com/WDCSecure/LabGNSS.git>