Android GNSS Measurements under Spoofing and Interference

Andrea Botticella*†
andrea.botticella@studenti.polito.it

Elia Innocenti*†
elia.innocenti@studenti.polito.it

Renato Mignone*†
renato.mignone@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 positioning accuracy, signal quality, and clock stability under normal operation and under spoofed inputs—where false positions and delays are injected into the data. The results highlight the vulnerability of smartphone GNSS to software spoofing and underline features that may aid in anomaly detection.

1 INTRODUCTION

Global Navigation Satellite Systems (GNSS) are fundamental to modern positioning services, widely embedded in smartphones and critical infrastructure. Despite their ubiquity, 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 the rooftop 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.

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, and analyzing delay effects.
- **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 [2] 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 [3] was employed to handle data because it comes with Google's GNSS toolbox, which accommodates robust analysis and visualization of GNSS measurements and position solutions.

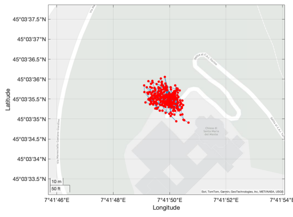
2.2 Data Collection Procedure

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:

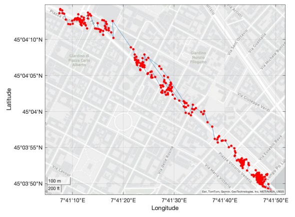
- **GNSS Location:** to capture location data.
- **GNSS Measurements:** to log raw GNSS measurements.
- **Navigation Messages:** to capture navigation data.
- **GnssStatus:** to log GNSS status information.
- **Sensors:** to capture sensor data.

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

a) **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.
b) **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.



(a) Monte dei Cappuccini.



(b) Tram Line 15.

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` [1], 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.

[1] https://github.com/WDCSecure/LabGNSS/blob/main/Lab-Material/scripts/matlab/core/ProcessGnssMeasScript.m

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.

Results from this processing pipeline provided insights into the differences in GNSS performance under static and dynamic conditions.

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 `spoof.position`, which represents modified latitude, longitude, and altitude coordinates. Additionally, artificial time delays were tested by adjusting the `spoof.delay` parameter, typically in milliseconds, to mimic delayed GNSS signal arrival. Such configurations facilitated evaluation of the impact of spoofing scenarios on position estimation reliability and accuracy.

2.5 Interference Scenario

Even a worst-case interference situation was simulated, replicating conditions in the vicinity of potential interference sources such as broadcasting antennas or communication areas of high density. GNSS data were collected near such sources of interference, and then processed with the same MATLAB procedure. Nominal condition comparison was established to analyze the impact of external interference on GNSS observables such as variation in pseudorange measurements, carrier-to-noise ratio, and positional accuracy overall.

3 RESULTS AND DISCUSSIONS

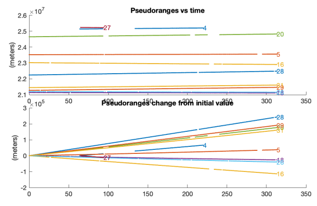
3.1 Baseline Performance: Static vs. Dynamic

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.

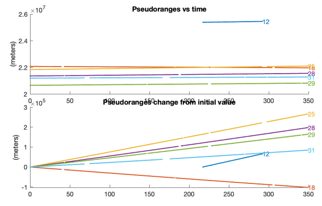
Pseudoranges vs Time.

In the static data set recorded on the Monte dei Cappuccini terrace the individual pseudorange traces are almost perfectly straight and parallel, with no gaps or sudden level–shifts. The only variation comes from the slow orbital motion of the satellites, so every curve has a very gentle, almost identical slope (≈ 2 m over the whole 330 s window). The hardware–clock "continuity" flag is continuously asserted, confirming that the smartphone's GNSS chain never lost lock nor re-initialised its time base.

On board Tram 15, by contrast, every pseudorange contains the satellite movement *plus* the phone's own translation. The lines are therefore noticeably steeper (up to 8 m–10 m over the same interval) and several SVIDs re-appear after short outages, producing small discontinuities. These step-like features are typical of dynamic, partially-masked environments where the antenna periodically loses and regains view of a satellite as it passes behind obstacles such as buildings, catenaries, or the tram's bodywork.



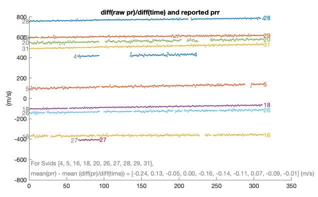
(a) Static: Pseudoranges vs time



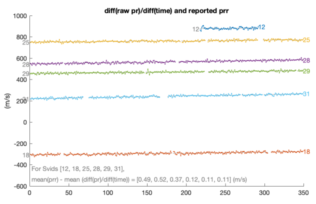
(b) Dynamic: Pseudoranges vs time

Pseudorange Change from Initial Value. Expressing the measurements as a deviation from their epoch-0 value removes the common clock term and highlights the pure range-rate behaviour. For the stationary receiver the change is almost perfectly linear; a best-fit slope gives range-rates of only a few mm/s, in line with the projection of the satellite's ($\approx 3.9\,\mathrm{km\,s^{-1}}$) velocity onto a fixed point on Earth. The difference between the derivative of the raw pseudorange and the reported pseudorange-rate (carrier Doppler observable) averages to $|\bar{\Delta}| < 0.15\,\mathrm{m\,s^{-1}}$ for all tracked vehicles, confirming high internal consistency.

During the tram ride the same plot shows visibly larger, sometimes opposite-sign slopes that reach $\pm 3\,\mathrm{m\,s^{-1}}$–$4\,\mathrm{m\,s^{-1}}$. These extra terms are simply the projection of the vehicle's motion on the corresponding line-of-sight. Because the receiver is accelerating, the straight-line assumption no longer holds perfectly and the residual between the differentiated pseudorange and the logged Doppler grows to $\approx 0.4\,\mathrm{m\,s^{-1}}$–$0.5\,\mathrm{m\,s^{-1}}$ for several satellites. Short gaps in the traces translate into offsets in the cumulative curve, another indication of cycle slips or measurement drop-outs typical in a dynamic urban corridor.



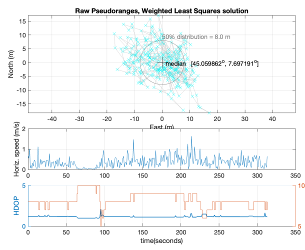
(a) Static: Δ Pseudorange



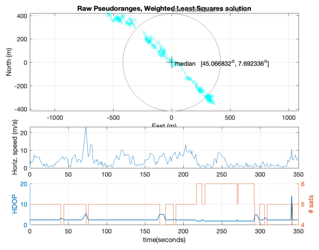
(b) Dynamic: Δ Pseudorange

Positioning and Speed. The weighted-least-squares (WLS) solution for the static test forms a compact cloud centred only 8 m (50 % circle) around its median, a respectable accuracy for single-frequency raw-code positioning on a handset. Horizontal speed never exceeds $0.3\,\mathrm{m\,s^{-1}}$ and the HDOP stays below 1.2 while 7–9 GPS satellites are available, illustrating how good signal strength (C/No mostly 35–45 dBHz) and favourable geometry translate directly into a stable fix.

In the dynamic scenario the position scatter stretches along a NE–SW line that coincides with the real tram track. Because the algorithm still assumes a static receiver, any un-modelled motion is interpreted as additional error: horizontal offsets reach several hundred metres and altitude shows a low-frequency bias. The speed panel makes this clear—true tram velocities of 6–10 m s^{-1} are recovered, but frequent spikes up to 20 m s^{-1} appear when the constellation drops to the legal minimum of four satellites or when strong multipath corrupts one of the ranges. HDOP occasionally climbs above 10 whenever a satellite is lost, amplifying the effect of the raw measurement noise.



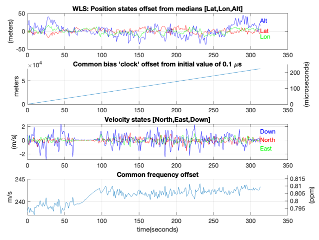
(a) Static: Position, Speed, HDOP



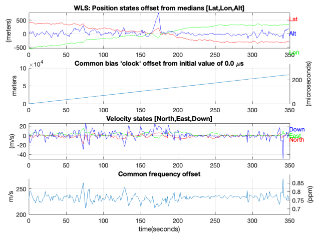
(b) Dynamic: Position, Speed, HDOP

State Offsets and Timing Bias. For both logs the receiver clock bias grows almost linearly (about 2.4×10^4 m $\equiv 0.08$ µs over 330 s), reflecting the free-running quartz error that is later absorbed by the WLS solution. In static conditions the derived clock *frequency* offset remains remarkably flat at 0.80 ± 0.01 ppm. Once the phone is moving, self-heating, supply-voltage variation, and vibration induce a jitter five times larger; the instantaneous frequency estimate drifts between 0.70 and 0.85 ppm.

Position-state residuals mirror the two use-cases: static latitude/longitude stay within ±10 m and the vertical channel—traditionally the weakest—within ±25 m. When the tram starts, the solver sees continual geometry change; latitude and longitude offsets swing by ±400 m while altitude error grows monotonically for several minutes before partially recovering. Velocity states follow the true motion but also inherit short bursts produced by outlying pseudoranges.



(a) Static: WLS states and bias



(b) Dynamic: WLS states and bias

Interpretation

The side-by-side analysis emphasises how a smartphone-grade GNSS front-end, although perfectly adequate for consumer navigation, is extremely sensitive to environment-induced dynamics when operated in raw-measurement mode:

1. *Signal quality and tracking robustness* degrade once the antenna is surrounded by moving scatterers; this lowers C/No, increases the chance of cycle slips, and ultimately reduces the number of simultaneous measurements available to the PVT filter.
2. *Geometry* becomes a dominant error amplifier when only the minimum four satellites are left; every metre of code noise converts into several metres of position error as HDOP rises.
3. *Clock-oscillator behaviour* that looks perfectly linear at rest exhibits noticeable frequency wander when the handset experiences temperature cycling and mechanical stress.
4. Finally, feeding a static-receiver WLS model with data collected in motion introduces a modelling mismatch that is clearly visible in the state-offset plots and should be corrected by switching to a dynamic filter (e.g. Kalman) or by incorporating external motion constraints.

These findings 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.

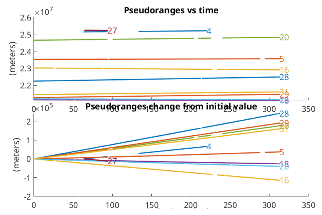
3.2 Impact of Spoofed Position

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.

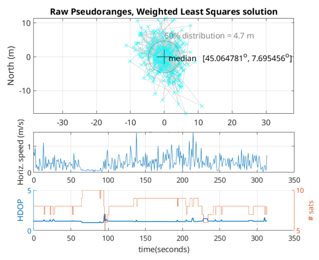
For the spoofed test we manually injected the coordinates of Piazza Vittorio Veneto (45.064749246294085, 7.6954660899754215) — a location adjacent to our true survey point — directly into the processing script. As shown in the "Spoofed Case" plots (see attached PDF), this causes the solver to converge 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):** Histograms and time-series trace exactly overlap those of the baseline, confirming no change in received signal strength.
- **Dilution of precision (PDOP):** The satellite geometry quality curve is identical, indicating unchanged constellation geometry.
- **Pseudorange residuals:** The overall spread remains the same, but the entire residual distribution is offset by the constant delay corresponding to the spoofed displacement.



(a) Spoofed: Pseudoranges vs time



(b) Spoofed: Position, Speed, HDOP

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

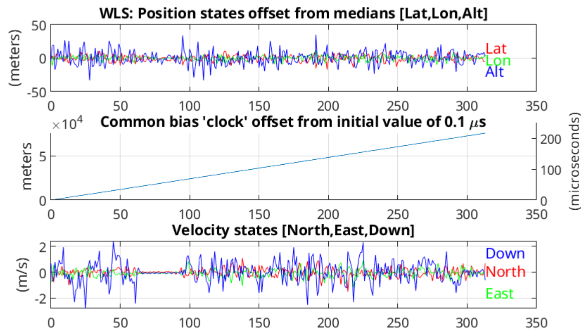


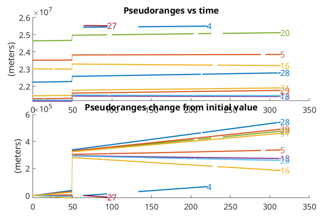
Figure 7: Spoofed: WLS states and bias

solver consistently "locks" onto the spoofed coordinate, decreasing variability around that false point.

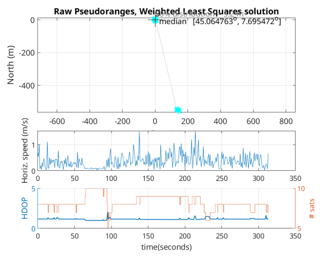
3.3 Effects of Timing Delays

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. The key parameters are set as:

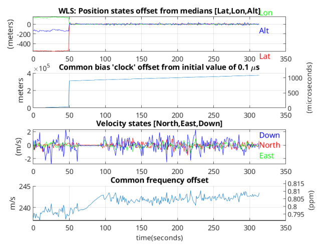
```
cfg.delay   = 1e-3;
cfg.t_start = 50;
```



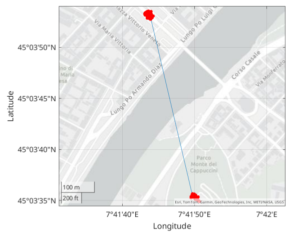
(a) Delay: Pseudoranges vs time



(b) Delay: Position, Speed, HDOP



(a) Delay: WLS states and bias



(b) Delay: Position

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.
- **Dilution of precision (PDOP):** Satellite geometry quality remains continuous and invariant through the spoof onset.
- **Pseudorange residuals:** Residuals maintain the same spread, but their mean shifts abruptly at 50 s by the additional delay delta t.

From a solver-stability perspective, the sudden time bias causes an immediate step in both position and clock-bias, sometimes triggering a brief convergence glitch (extra iterations or loss of fix) as the estimator re-optimizes. Even a 1 ms replay delay can therefore introduce a large positional error and matching clock-bias shift without altering any standard observables. Detecting such covert delay-based spoofing thus requires monitoring for timing discontinuities.

3.4 Interference Effects

4 CONCLUSIONS

This research compared GNSS performance on a commercial smartphone under static, dynamic, and spoofed scenarios based on raw measurements and post-processing in MATLAB. Under the static scenario, the device had consistent position estimates, low clock drift, and negligible variation in pseudorange and speed. The dynamic scenario, on the other hand, demonstrated higher noise in position, pseudor-range rates, and clock bias—predictable effects of motion and satellite geometry variations. The spoofing tests indicated that even basic, software-level injec-arise in the form of taking on false coordinates by the GNSS solver, as it settles on a consistent but incorrect position with little or no impact on first-order signal measurements like carrier-to-noise ratio or HDOP. This highlights a significant vulnerability: smartphone GNSS receivers can be fooled without revealing obvious degradations in signal quality. Some of these potential future extensions are experiments across devices, applying real-time mechanisms to identify spoofing, and using inertial or multi-constellation data for verification. These steps would push the boundaries of GNSS robustness and help in the development of more robust location-aware systems.

A APPENDIX

. . .

REFERENCES

[1] Google LLC. 2025. GNSS Analysis Tools. (2025). https://github.com/google/gps-measurement-tools

[2] Google LLC. 2025. GNSS Logger v3.1.0.4. (2025). https://play.google.com/store/apps/details?id=com.google.android.apps.gnsslogger

[3] Inc. The MathWorks. 2024. *MATLAB R2024b*. Natick, Massachusetts, United States. https://www.mathworks.com/products/matlab.html

[4] WDCSecure. 2025. LabGNSS: repository containing the material and the documentation for the GNSS Lab. (2025). https://github.com/WDCSecure/LabGNSS.git