GPS Solutions (2024) 28:167
https://doi.org/10.1007/s10291-024-01705-8

ORIGINAL ARTICLE

Real-time cloud computing of GNSS measurements from smartphones and mobile devices for enhanced positioning and navigation

Jorge Hernández Olcina[1] · Ana B. Anquela Julián[1] · Ángel E. Martín Furones[1]

Received: 14 May 2024 / Accepted: 11 July 2024

Abstract
In recent years, Global Navigation Satellite Systems (GNSSs) have become integral to our daily lives because of their precise positioning and navigation capabilities. Widespread use of smartphones equipped with GNSS receivers results in the generation of a huge amount of positioning data. Therefore, real-time cloud computing has emerged as a promising approach to effectively leverage this wealth of location information. In this study, we developed an Android app that captures raw GNSS data from smartphones, leverages cloud computing resources, calculates the position of the device, and returns the computed solution to the user. Integration of cloud-based processing not only conserves the device resources but also enables real-time position calculation, paving the way for enhanced location-based applications and services.

Keywords Real-time GNSS · Smartphone · Cloud computing · Android

Introduction

The advent of smartphones with built-in GNSS receivers has revolutionized positioning technology, enabling users to access accurate location information on a global scale. The development of low-cost GNSS chips has spurred a revolution in positioning and navigation devices. The abundance of GNSS measurements from smartphones has opened new opportunities for various applications, including location-based services, transportation management, disaster response, and environmental monitoring (Banville et al. 2016).

During the "I/O 2016" conference in May 2016, Google made a significant announcement regarding the availability of raw GNSS measurements from Android smartphones and tablets running the Android N operating system, also known as "Nougat", version 7 (Banville et al. 2016; GSA 2016). Subsequently, in May 2018, the market witnessed the introduction of the first multi-constellation and double-frequency mobile phone, Xiaomi MI 8 (Robustelli et al. 2019). These developments represent major milestones in the new era of positioning and navigation. The implications of these advancements are of great significance to the scientific community as they signify a departure from the conventional GNSS receiver black-box concept, which previously provided metric-level accuracies. With access to Pseudorange, Doppler, and Carrier Phase measurements, the possibility of obtaining more accurate positions has increased (Banville et al. 2016). These breakthroughs have the potential to revolutionise the fields of positioning and navigation, thereby providing new opportunities for enhanced accuracy and performance.

Currently, the GNSS technology is facing unprecedented challenges driven by the ever-increasing demand for flawless, accurate, and reliable positioning solutions. The emergence of new applications has further compounded this situation, as they require smaller form factors, lower energy consumption, and constrained computational capabilities for user devices. These factors raise serious concerns regarding the implementation of GNSS data-processing tasks (García-Molina et al. 2017; Lucas-Sabola et al. 2016). Striking a balance between meeting the rigorous demands for precise positioning and accommodating the constraints of modern applications is a significant hurdle.

The rise of mobile technology has dramatically transformed GNSS research. Smartphones and other common devices with built-in GNSS capabilities have opened exciting new possibilities for applications like real-time

✉ Jorge Hernández Olcina
jorgeho1995@gmail.com; jorherol@doctor.upv.es

[1] Department of Cartographic Engineering, Geodesy and Photogrammetry, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

Published online: 27 July 2024

kinematics (RTK) and precise point positioning (PPP). This is particularly significant for navigation and positioning solutions, as mobile devices offer widespread availability and cost-effectiveness.

Recent research underlines this potential. For example, a study by Mahato et al. (2024a, b) successfully employed a CLS GNSS module and open-source software for long-distance RTK applications in India (single baseline). Their findings demonstrated promising accuracy and reliability, paving the way for these systems in various practical scenarios. Additionally, research has extensively explored using regular Android smartphones with dedicated apps for cost-efficient GNSS data collection. Mahato et al. (2024a, b) again highlighted the effectiveness of these tools, emphasizing their affordability and ease of use, which can significantly reduce barriers to GNSS research and applications.

Dutta et al. (2020) further explored the capabilities of Android smartphones in a multi-constellation environment. Their findings confirm that these devices can handle complex GNSS studies, expanding the versatility and scope of mobile GNSS applications. This research is crucial for understanding the practicalities and limitations of using consumer-grade devices for high-precision GNSS data collection and analysis.

The need for accurate, precise, and safe positioning solutions has pushed receiver technology to its limits, requiring innovative approaches to optimise the performance in constrained environments. The efficient processing of GNSS data while considering size, energy, and computational limitations is now a critical aspect in advancing positioning and navigation technologies. Overcoming these challenges will play a pivotal role in shaping the future of GNSS applications and meeting the diverse needs of users across various domains. Hence, the need to address this challenge efficiently has given rise to the concept of leveraging cloud computing (Favenza et al. 2014; García-Molina et al. 2017; Lucas-Sabola et al. 2016; Lucas-Sabola et al. 2018). By embracing cloud computing, obstacles associated with GNSS data treatment and processing within receivers can be overcome. Additionally, this approach offers a solution to the issue of energy consumption because many GNSS receivers rely on batteries that deplete faster with increased computational loads (Lucas-Sabola et al. 2016). Cloud computing provides a promising avenue for offloading computation-intensive tasks to remote servers, thereby reducing the burden on local devices and optimising energy utilisation. This change in thinking opens new possibilities for more resource-efficient and sustainable positioning and navigation solutions in the GNSS technology realm.

Studies by Konstantinos et al. (2013) and Liu et al. (2021) advocate the implementation of cloud computing for geospatial data processing. Konstantinos et al. (2013) proposed a client–server structure to handle geospatial data, which can be adapted to the scenario explored in this study. The client is a smartphone that captures GNSS data and transmits them to the server for processing, resulting in the calculation of the position of the smartphone. Liu et al. (2021) followed a similar approach, focusing on RTK positioning. This suggests an algorithm for data capture and processing on the server side, enabling precise calculation of positions based on an accurate capture time. Both studies demonstrated the potential of cloud computing to enhance geospatial data processing, opening new avenues for achieving more accurate and efficient positioning solutions. By leveraging cloud resources, these approaches address the challenges posed by limited client-side capabilities and demonstrate the transformative impact of cloud-based methodologies in geospatial applications.

Therefore, this study proposes an innovative approach that combines raw GNSS data from Android devices with cloud computing to achieve higher location accuracy. An Android application was developed to utilise the GNSS API (Application Programming Interface) to access raw measurements, which were then processed in the cloud, leveraging vast computational resources for improved positioning algorithms. This integration promises groundbreaking advancements in the accuracy and efficiency of mobile device apps. Furthermore, the establishment of a cloud-based data store unlocks possibilities for future analysis, such as studying the behaviour of the troposphere or ionosphere and leveraging the wealth of data collected from smartphones.

Methodology

This section discusses the methodologies employed to develop the app. The explanation is presented in two distinct phases: first, a detailed breakdown of each constituent part, encompassing the front-end, back-end, and other elements; and second, a comprehensive overview of the workflow entailed in calculating positions through cloud computing.

Frontend

An Android app was developed using the Flutter framework (Flutter 2023). As an open-source mobile app development SDK (Software Development Kit) developed by Google, Flutter has garnered widespread popularity because of its versatility and efficiency in crafting high-quality user interfaces. Providing a seamless experience for Android, iOS, and web applications has become the preferred choice for developers aiming to create visually appealing and responsive cross-platform applications.

The app delineates two principal functionalities: first, an offline logger-type capability facilitating data capture and storage in plain text files poised for subsequent

Fig. 1 Illustration of a raw data log file: Capturing data fields in Log Mode and exporting to a text file. Header presents each stored value

post-processing (see Fig. 1), and second, an online feature enabling real-time processing of raw data within the cloud environment, representing the study's central objective. In both scenarios, direct access to the raw data sourced from the smartphone's GNSS sensor was established using the Android API (Android 2016). From the raw data, a satellite message was generated, conforming to the identical structure implemented in Google's *GNSSLogger* app (Android 2016). Adopting a format identical to that of the Google app ensures compatibility with the analysis tools provided by Google to users, thereby enabling seamless utilisation of raw GNSS data.

In summary, the app serves as a GNSS receiver, performing dual functions of data transmission by either storing data within files or forwarding it to the backend for processing. In a real-time context, the app receives positional outcomes computed at the server end and subsequently retains and presents these outcomes to the user.

Backend

On the server side, the backend infrastructure is constructed using NodeJS (NodeJS 2023). This cross-platform open-source runtime environment was developed in the JavaScript programming language. Noteworthy for its asynchronous and event-driven architecture, NodeJS is well suited for handling data I/O in server-layer applications. Fuelled by Google's V8 engine, NodeJS exhibits a remarkable performance, making it an excellent foundation for building scalable and efficient server-side solutions. Its extensive range of libraries and packages offers developers a robust ecosystem for streamlining application development and optimising performance. However, the NodeJS single-thread architecture may impact the scalability of the solution. Detailed discussions on multiple user concurrency and scalability, which are beyond the scope of this study, are addressed in "Multiple user concurrency" section as part of future work.

For real-time positioning computation, the RTKLib calculation engine is harnessed, which is a powerful open-source software package (RTKLib 2013). Designed to facilitate standard and accurate positioning, it comprises a portable program library and various application programs, all leveraging GNSS technologies. The versatility of the library allows for the tailoring of custom applications and solutions to a wide range of GNSS-related tasks, making it a valuable tool for GNSS data processing requirements.

Finally, to enhance the speed of the initial epoch calculation, the BKG NTRIP Client (BNC) tool was employed (BNC 2023). This is a specialised software tool developed by the German Federal Agency for Cartography and Geodesy (BKG) for accessing and utilising the networked transport of the Radio Technical Commission for Maritime Services (RTCM) via Internet Protocol (NTRIP) data streams. NTRIP is used to transmit GNSS correction data and other positioning-related information over the Internet in real time. The BNC provides a user-friendly interface for connecting to NTRIP caster servers and receiving real-time GNSS observations, navigation and clock messages, and correction data streams from Continuously Operating Reference Stations (CORS) and other GNSS infrastructures. Correction data streams are crucial for enhancing the accuracy and precision of GNSS positioning, particularly in applications such as

real-time kinematics (RTK) positioning and precise point positioning (PPP). Specifically, the BNC was configured to connect to an NTRIP caster, receiving a Mount Point that exclusively transmits navigation messages. These messages are then forwarded through a port, where a NodeJS application listens for them and subsequently sends them to RTKLib for further processing.

WebSocket connection

The communication linkage between the app and server, as depicted in Fig. 2, has been established through the utilisation of the WebSocket network protocol which is considered the best method for real-time applications owing to its numerous advantages and unique characteristics (WebSockets 2023).

Here are some reasons why WebSockets are the preferred choice for real-time communication (WebSockets 2023):

- *Low Latency and Bi-Directional Communication*: WebSockets facilitate low-latency communication between the client and the server, enabling real-time data exchange in both directions. Unlike traditional HTTP requests in which the client initiates communication and the server responds, WebSockets enable ongoing full-duplex communication, allowing data to be sent and received in real time without delay.
- *Efficient and Lightweight*: WebSockets are lightweight and require minimal overhead to establish and maintain connections. Once the initial connection is established, subsequent data transfers occur through the same connection, thereby reducing the overhead of repeatedly establishing new connections for each data exchange.
- *Continuous Connection*: With WebSockets, the connection between the client and server remains open for as long as required. This persistent connection ensures that data can be sent and received instantly, making it suitable for real-time applications requiring frequent updates and rapid response times.
- *Real-Time Updates*: WebSockets enable real-time updates, making them ideal for applications that require live data streams such as chat applications, collaborative tools, real-time gaming, financial platforms, and location-tracking systems.
- *Scalability*: WebSockets support concurrent connections, which makes them highly scalable. Servers can efficiently handle many simultaneous connections without sacrificing performance, making them suitable for applications with many concurrent users.
- *Event-Driven Architecture*: WebSockets are built on an event-driven architecture, allowing servers to push data

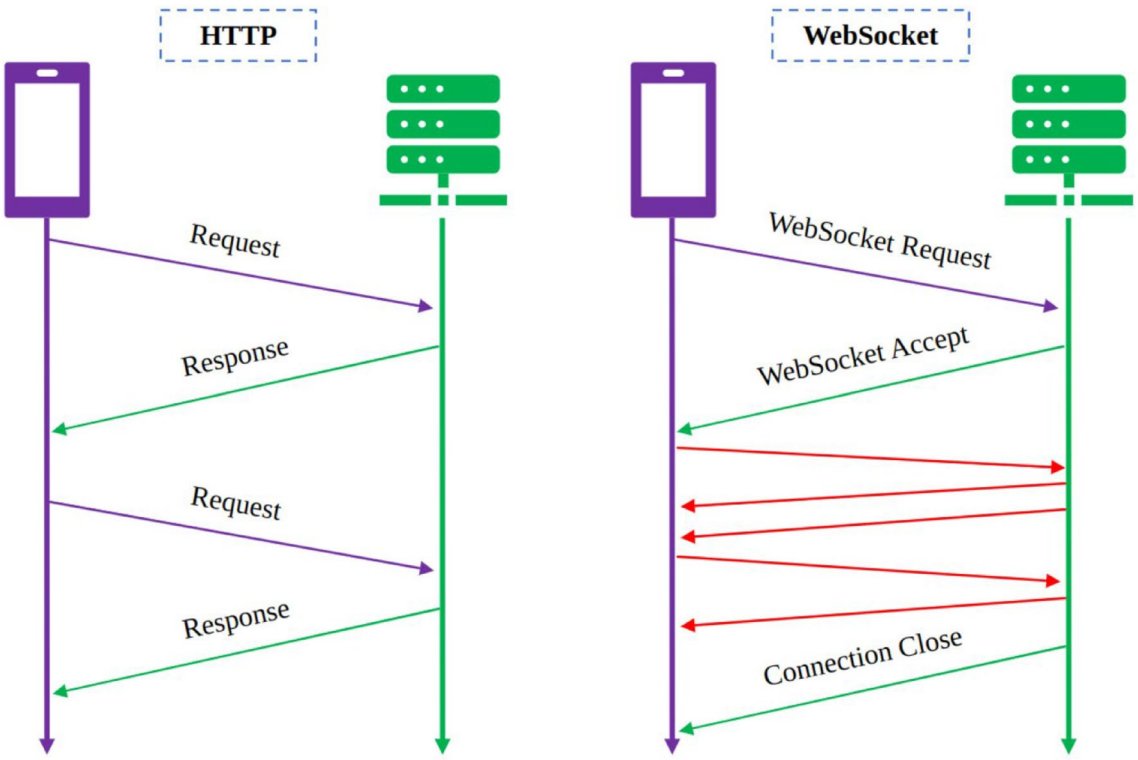


Fig. 2 Comparative Diagram: HTTP versus WebSocket Connection. As depicted in the diagram, WebSocket establishes a connection wherein both parties can exchange messages independently after initialization, eliminating the need to await the other party's response

to clients when specific events occur, thereby eliminating the need for constant polling.

- *Security*: WebSockets can be secured using standard encryption protocols such as SSL/TLS, ensuring that data transmitted over the connection remain secure and protected from eavesdropping or tampering.

In conclusion, the adoption of WebSockets is an effective and dependable approach for real-time communication between clients and servers. Their attributes include low latency, bidirectional capabilities, lightweight architecture, and extensive browser compatibility, rendering WebSockets optimal for constructing responsive and interactive real-time applications that span diverse sectors and scenarios. This suitability extends the focus of the present study.

GNSS cloud computing workflow

In this integrated system, as shown in Fig. 3, the Android app plays a significant role by serving as a data source for the GNSS sensor integrated into a smartphone. The app captures raw GNSS data, including satellite Pseudoranges, Carrier phases (if available), and Doppler measurements, with unrestricted access to the smartphone's GNSS sensor. Once the relevant GNSS data are collected, the app establishes a WebSocket connection to the server to facilitate efficient data transmission (see "WebSocket connection" section. WebSocket Connection.)

Upon establishing the connection between the app and the server, the app initiates the process by transmitting a "start" message. This message signals the server to commence preparations for execution, whereas the app enters a state of readiness to accept a notification, indicating its

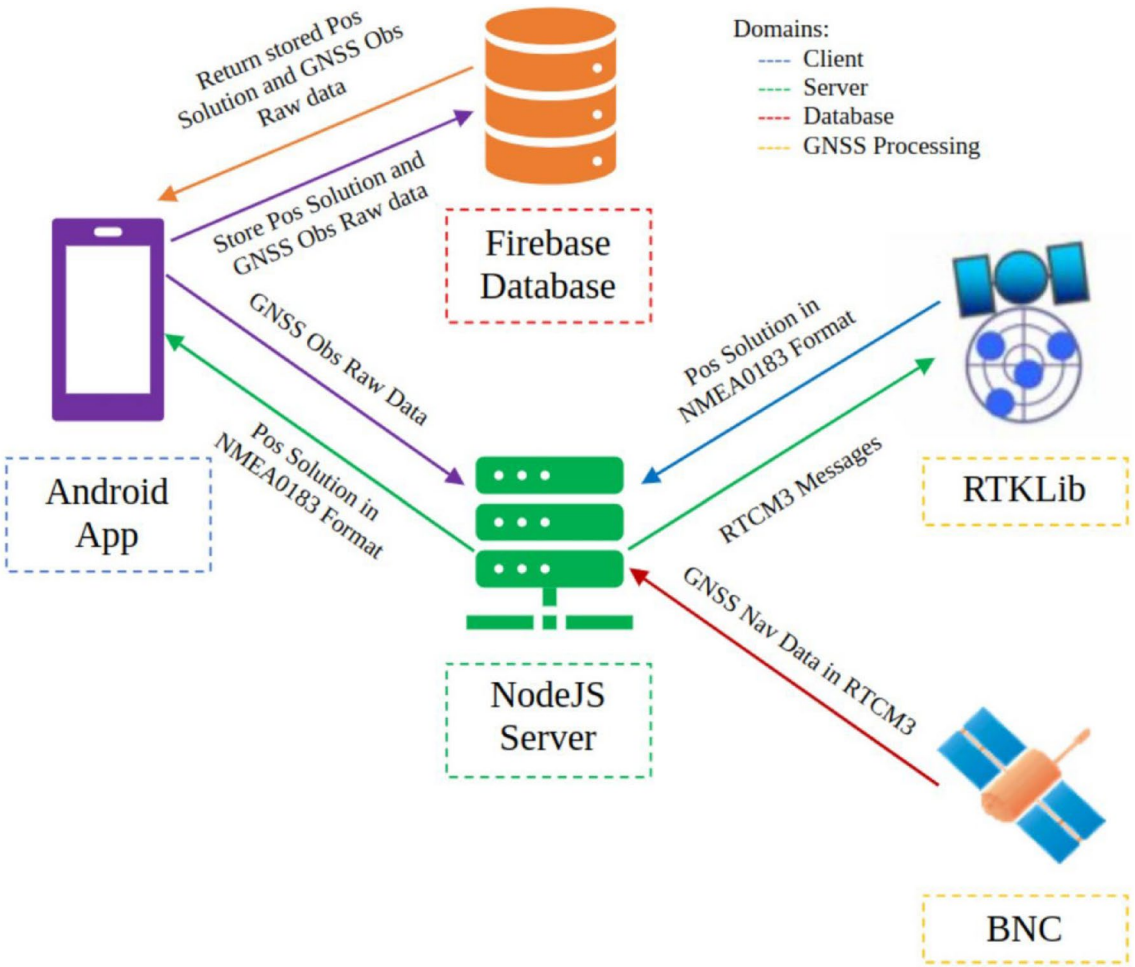


Fig. 3 Workflow within the Cloud Computing platform is characterized by five pivotal components: (i) Android app, serving as a global navigation satellite system (GNSS) receiver, (ii) NodeJS server, overseeing real-time executions, (iii) BKG NTRIP Client (BNC), responsible for navigation message capture, (iv) RTKLib, functioning as the positioning calculation engine, and (v) Firebase, serving as a database management system

preparedness to collect and process raw GNSS data. In the initial steps, the server triggers the launch of an RTKLib instance, simultaneously establishing a bidirectional TCP/IP connection for seamless data exchange. This connection ensures the server's ability to seamlessly transmit and receive data. Another TCP/IP connection is established with an ongoing instance of the BNC persistently running in the background. This connection enables the server to retrieve navigation messages. With these essential components in place, the server communicates with the app that is primed to accept messages housing the raw data from the GNSS sensor. This synchronisation provides a framework for effective data reception and processing.

As soon as the raw GNSS data are received, the server performs a crucial preprocessing step by converting the data into the RTCM3 (Radio Technical Commission for Maritime Services—Version 3) format, which is widely used for real-time differential GNSS applications. This data preparation is essential for subsequent delivery to the RTKLib software. RTKLib was the core processing engine used in this system. RTCM3 is a standard data format used to transmit GNSS observations and navigation data, corrections, and other auxiliary information over communication channels in real-time GNSS applications (RTCM 2013). This is commonly employed to improve the accuracy of GNSS positioning.

For this conversion to RTCM3 format, a sequence of computations is required to derive the Pseudorange, Carrier Phase, Doppler measurements, and temporal values. As elucidated by the GSA (2016), given the unavailability of these measurements directly from the Android API, their determination requires the following methodology (Android 2016; GSA 2016):

GPS Time Generation: In context of Android 7 or higher, the direct provision of GNSS time is unavailable. However, an internal hardware clock and bias towards the true GPS time (expressed in nanoseconds) are offered, provided that the receiver has gauged the GNSS reference time. In instances where the receiver ascertains the GPS time, the computation can be conducted as follows:

$$GnssTime = TimeNanos - (FullBiasNanos + BiasNanos)[ns] \quad (1)$$

where *TimeNanos* is the value of the GNSS receiver's internal hardware clock, expressed in nanoseconds; *FullBiasNanos* signifies the bias between the receiver's clock and GPS time in nanoseconds; and *BiasNanos* is the sub-nanosecond bias of the clock. If the receiver has inferred time via a non-GPS constellation, the calculated GPS time can be derived using the following formula:

$$GpsTime = TimeNanos - (FullBiasNanos + BiasNanos) - InterSystemsBias \quad (2)$$

where *InterSystemsBias* denotes the disparity between the GPS and GNSS times used in the time estimation process. For instance, if the Galileo System Time is employed for time estimation, *InterSystemsBias* is represented by the GPS to Galileo time offset (GGTO).

Pseudorange Generation: The Android system does not provide direct pseudoranges but offers all the essential parameters for their computation. The formulation of pseudoranges hinges on temporal disparity, specifically the time lapse between the instances of reception (measurement time) and transmission.

$$\rho = \frac{(t_{Rx} - t_{Tx})}{1E9} \cdot c \quad (3)$$

where t_{Tx} represents the received GNSS satellite time at the measurement juncture or the emission time, which corresponds to the GNSS reference time at which the signal is dispatched. t_{Rx} denotes the measurement time or received time, and c symbolizes the velocity of light in vacuum. t_{Tx} is furnished by the Android system and is structured as

$$t_{Tx} = ReceivedSvTimeNanos[ns] \quad (4)$$

where *ReceivedSvTimeNanos* denotes the emission time in ns.

The acceptable scope of t_{Rx} can be reconstructed in the form:

$$t_{Rx_{GNSS}} = GnssTime + TimeOffsetNanos \quad (5)$$

Here, *TimeOffsetNanos* signifies the time offset at which the measurement was captured, denoted in nanoseconds. This value specifies the measurement timing accuracy.

Carrier Phase Measurements: Android supplies carrier phase measurements as *AccumulatedDeltaRangeMeters* (ADRM) represented in meters.

These measurements are inherently ambiguous and lack temporal information, indicating that the receiver can only quantify the cycles that occur between epochs. In the event of a cycle slip, this count was forfeited. The veracity of the carrier measurements is indicated through the *AccumulatedDeltaRangeState*(ADRS) field, which offers several flags detailed in Table 1. For the computational process, only valid measurements in this category were considered.

Doppler Measurements: The Doppler shift arising from satellite motion can be deduced from the *PseudorangeRateMetersPerSecond* parameter, provided that the pseudorange rate registered at the timestamp or received time is in meters per second (m/s). Notably, this value, presented in Hertz (Hz), does not encompass adjustments for receiver and satellite clock frequency discrepancies, making it an "uncorrected" value. A positive

Table 1 AccumulatedDeltaRangeState (Source Android 2016; GSA 2016)

ADR state	Value	Description
UNKNOWN	0	The state is invalid or unknown
VALID	1	The state is valid
RESET	2	A reset is detected
CYCLE_SLIP	4	A cycle slip is detected
HALF_CYCLE_RESOLVED	8	Half cycle ambiguity is resolved at time t
HALF_CYCLE_REPORTED	16	Half cycle ambiguity is reported at time t

"uncorrected" value signifies the satellite's motion away from the receiver. The connection between the "uncorrected", "pseudorange rate" and the "doppler shift" is articulated through the equation:

$$doppler = -\frac{PseudorangeRateMetersPerSecond}{k} \tag{6}$$

Here, k is a constant contingent on the centre frequency of the signal, for instance, f_c for L1 at 1575.42e6 Hz and the speed of light c. Thus, k is represented by c/f_c, which denotes the wavelength.

Once the previously delineated computations were executed, the transmission of the GNSS raw data to RTKLib was facilitated through the utilisation of the MSN5 (Multiple Signal Messages—Type 5) message from the RTCM3 format, which is designed to carry satellite observations from multiple constellations and frequencies. Tables 2, 3, and 4 present the predominant MSM5 message fields along with their corresponding values or formulas extracted from the raw GNSS data.

MSM5 message contains detailed information about the satellite measurements, including the Pseudorange, Carrier Phase, Doppler measurements, and signal quality indicators. These observations were collected from various GNSS satellites including GPS, GLONASS, Galileo, Beidou, and other regional satellite systems.

With all the components and connections in place, the server transmits the converted RTCM3 messages from the raw GNSS data and the necessary navigational information to RTKLib, which initiates its computations to generate a precise positioning solution. When RTKLib successfully calculated the solution, it was transmitted back to the server, which was then collected and forwarded to a smartphone. The smartphone stores the solution in plain text files, presenting the outcome to users. In terms of the data format, the information procured from RTKLib adheres to the NMEA0183 format (National Marine Electronics Association), which includes the messages defined in Table 5.

To ensure data integrity and reliability, the smartphone securely stored files in the Firebase database to create a cloud-based data store for future analysis. Firebase is a Google powered mobile and web application development platform that offers a range of tools and services (Firebase 2023). This simplifies the development process by providing a real-time database, authentication, cloud functions, and

Table 2 Content of the message header for MSM5

Data field	Value/formula
GNSS epoch time	*GpsTime* from (1) or (2)
Multiple message bit	*True*, more than one constellation *False*, one constellation

Table 3 Content of satellite data for MSM5

Data field	Value/formula	
Number of integer milliseconds in GNSS satellite rough ranges	$Round\left(\frac{\rho \cdot 2^{10}}{c \cdot 10^{-3}}\right) \gg 10$	(7)
GNSS satellite rough ranges modulo 1 ms	$Round\left(\frac{\rho \cdot 2^{10}}{c \cdot 10^{-3}}\right) \& 1023$	(8)
GNSS satellite rough PhaseRangeRates	$Round\left(\frac{doppler \cdot c}{CarrierFreqHz}\right)$	(9)

Table 4 Content of signal data for MSM5

Data field	Value/formula	
GNSS signal fine pseudoranges	$\left(\rho - \left(Round\left(\frac{\rho \cdot 2^{10}}{c \cdot 10^{-3}}\right) \cdot \frac{c \cdot 10^{-3}}{2^{10}}\right)\right) \cdot \frac{2^{24}}{c \cdot 10^{-3}}$	(10)
GNSS signal fine PhaseRange data	$\frac{Round\left(ADRM - \left(Round\left(\frac{\rho \cdot 2^{10}}{c \cdot 10^{-3}}\right) \cdot \frac{c \cdot 10^{-3}}{2^{10}}\right)\right) \cdot 2^{29}}{c \cdot 10^{-3}}$	(11)
GNSS signal CNRs	$Round(Cn0DbHz)$	(12)
GNSS signal fine PhaseRangeRates	$Round\left(\frac{\left(-\frac{doppler \cdot c}{CarrierFreqHz} - Round\left(\frac{doppler \cdot c}{CarrierFreqHz}\right)\right)}{10^{-4}}\right)$	(13)

Table 5 Content of NMEA0183 messages from RTKLib solution output

Message	Function
RMC	Position, velocity, and time
GGA	Time, position, and fix related data
GSA	GNSS DOP and active satellites
GSV	Number of SVs, PRN, elevation, azimuth, and SNR

hosting. Firebase is known for its ease of use, scalability, and seamless integration with other Google services, making it a popular choice for developers to build apps of any size. Therefore, the Firebase database provides a seamless and efficient way to store and manage the positioning solutions obtained from RTKLib processing and raw GNSS data. Main features from Firebase used in this research include:

- *User Authentication*: This component verifies the identity of application users attempting to access the system. This ensures only authorized individuals can interact with the data.
- *Real-time Database*: This functionality facilitates the storage of information pertaining to app executions. This includes data capture, processing, and any associated timestamps, allowing for real-time analysis and monitoring.
- *Storage*: This component provides persistent storage for raw GNSS data and processed NMEA messages containing positioning solutions. These text files serve as the primary data source for further analysis or archival purposes.

Latency

Latency has emerged as a pivotal concern in the application of this concept. Notably, such executions inherently operate within a quasi-real-time paradigm, as a perceptible delay is inevitable in a sequence involving message reception, transmission to the server, subsequent processing, and eventual solution reception. The focus of this study is to delineate the methodological approach used to institute the entire infrastructure.

To assess the impact of latency on positioning accuracy, an extensive investigation was conducted involving two types of tests: a four-hour static test and two fifteen-minute kinematic tests (one involving pedestrian movement and the other involving vehicles). The static test aimed to evaluate long-term operational conditions, focusing on how delays in navigation message updates due to outdated ephemeris data affect positional accuracy. In contrast, the kinematic tests were designed to capture real-time dynamics, examining the feasibility of the technology for real-time applications where immediate positioning solutions are essential. The findings from these tests provide a detailed understanding of the behaviour of latency and its influence on positioning accuracy, offering valuable insights for the practical implementation of cloud-based GNSS applications (Hernández Olcina et al. 2024).

Multiple user concurrency

At the current stage of development, implementation has been intentionally conducted on a server with limited resources to focus on evaluating the benefits of the methodology. This approach allows for the controlled analysis of a system's capabilities and performance under constrained conditions.

However, it is important to recognise that scaling-up to accommodate concurrent users is a critical consideration for applications of this type. As the user demand increases, a server's limited resources can become a bottleneck, potentially leading to decreased performance and responsiveness. Several strategic solutions have been explored to address this challenge.

- *Cloud Infrastructure*: Transitioning to cloud platforms, such as Google Cloud or AWS, offers the advantage of scalability. These platforms allow for the dynamic allocation of resources based on demand, ensuring that the system can handle varying levels of user concurrency without compromising performance.
- *Instance Isolation*: The utilisation of dedicated instances for individual user interactions can prevent contention and resource sharing issues. This isolation ensures that the performance of one user's interaction does not affect that of others.
- *Containerization and Orchestration*: Employing technologies such as Docker and Kubernetes enables efficient management of execution queues and resource allocation. Containerisation enables the encapsulation of application components with their dependencies, ensuring consistent behaviour across different environments. Kubernetes helps automate the deployment, scaling, and management of containerised applications, enabling the streamlined management of concurrent executions.
- *Load Balancing*: Load balancers can distribute incoming user requests across multiple servers or instances, preventing any single resource from becoming overwhelmed. This approach helps maintain system responsiveness and prevents overutilisation of specific resources.
- *NodeJS single thread architecture*: While NodeJS utilizes a single-threaded event loop, its scalability can be evaluated by measuring the number of concurrent connections it can efficiently handle before performance degradation. A future work direction could involve designing a load testing framework that simulates a steadily increasing number of persistent connections, while monitoring response times and resource utilization (CPU, memory) to determine the point at which NodeJS exhibits a significant drop in performance. This would provide valuable

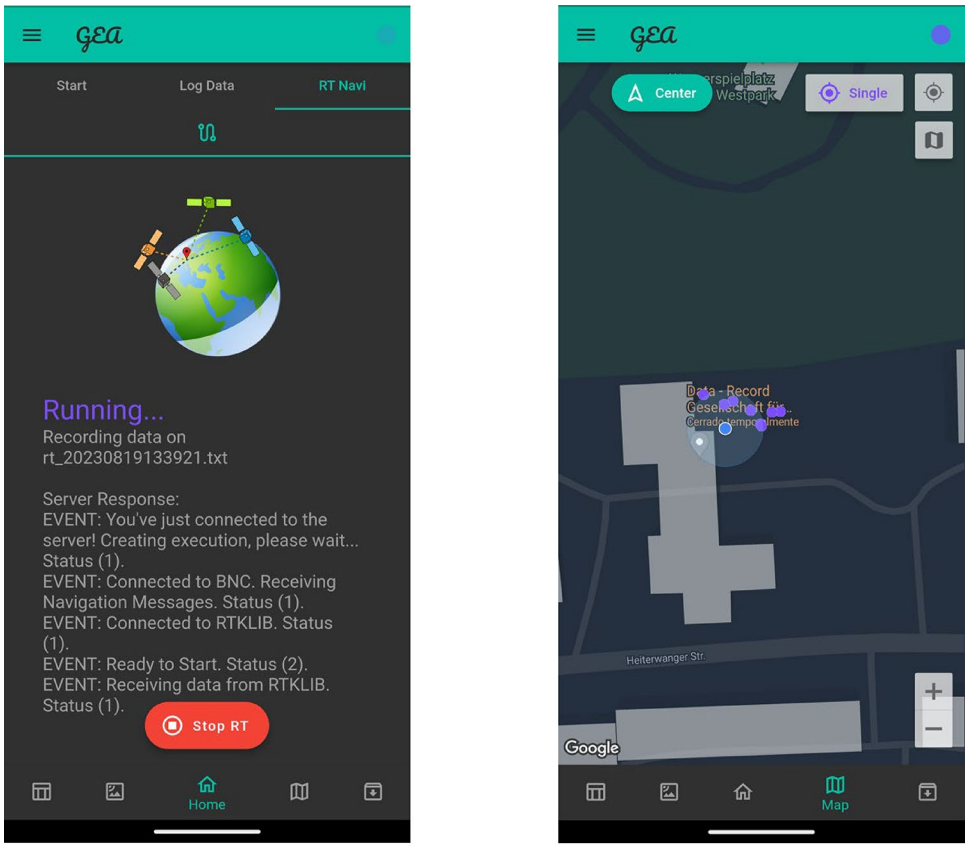


Fig. 4 Interface examples 1. **a** Real Time Launcher tab displays the real-time execution status. It provides a visual representation of the connection's current state with the server during an ongoing RT execution. **b** Map tab exhibits the real-time positional solution achieved via single processing. This tab visually represents the instantaneous position obtained during the ongoing execution. Maps generated using GEA application version 2.1.0

insights into the practical scalability limitations of Node.js for real-world applications.

Incorporating these strategies enables the application to effectively manage the challenges posed by concurrent user interactions. By combining a scalable cloud infrastructure, containerisation, load balancing, and careful resource management, the system can provide a seamless experience to users while efficiently utilising available resources.

Results

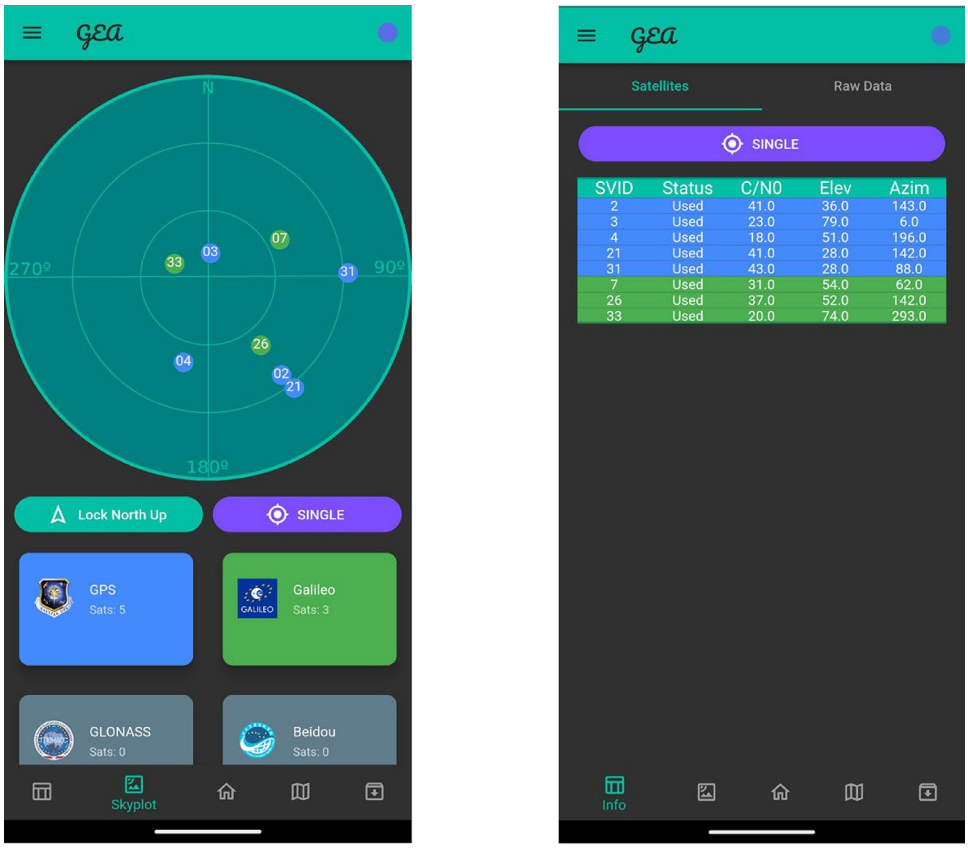
This section presents the results of the implementation detailed in "Methodology" section. It encompasses the final user interface and outlines a real-world field experiment conducted to assess the performance and behaviour of the app. The design, functionality, and real-time behaviour of the app's interface was examined to validate its practicality and effectiveness.

App Interface

Figures 4 and 5 depict the primary interface components during the real-time execution. Upon receipt of the positioning solution, the app not only archives these data but also presents them within the app for analytical assessment. These figures highlight the pivotal elements of the interface, while displaying the real-time execution process and subsequent data analysis within the app.

In scenarios where no real-time execution is in progress, or when data is being captured in *"Log"* mode, the *"Sky-plot"* and *"Info"* tabs transition to highlighting satellite data sourced from the *GnssStatus* interface of the Android API (Android 2016). This interface conveys the satellite elevation and azimuth details, along with their C/No values. As illustrated in Fig. 6, this feature empowers users to conduct a preliminary analysis of satellite visibility prior to real-time execution. It offers insights into satellite visibility patterns and facilitates the examination of the received frequencies for each satellite system, particularly in the context of dual-frequency devices.

Both the app development and subsequent testing were conducted using the Google Pixel 7 Pro smartphone as the



(a) (b)

Fig. 5 Interface examples 2. **a** Skyplot tab displays the satellite constellation's arrangement in the sky during the position acquisition process. **b** Info tab provides a tabulated display of satellites, accompanied by their corresponding elevation, azimuth, and carrier-to-noise ratio (C/No) values

primary work tool. This choice stems from its exceptional capabilities, particularly in terms of accessing GNSS sensor data without hardware restrictions. Google Pixel 7 Pro offers a comprehensive and user-friendly interface for retrieving crucial GNSS information, making it an ideal platform for conducting experiments and validating the proposed solution.

Experiment setup and results

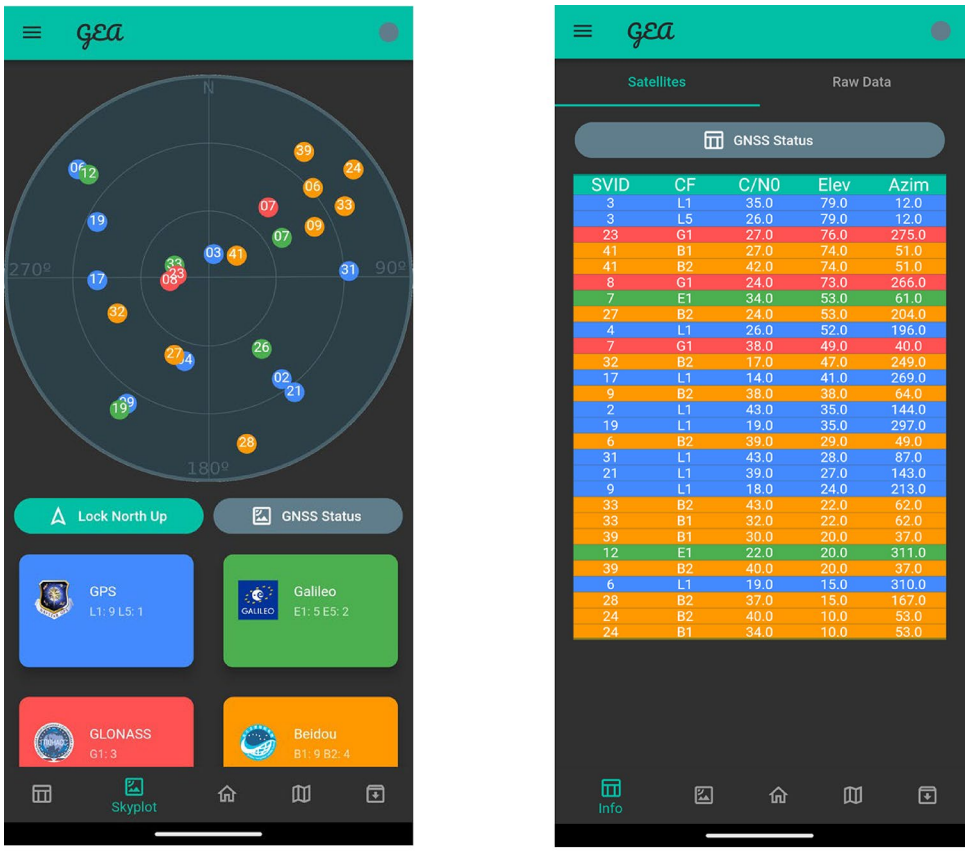
To evaluate the performance of the app, a field test was conducted under real-world conditions using the following configuration:

The primary aim of the test was to assess the performance of the cloud computing solution, rather than the positioning solution itself. Using the hardware setup outlined in Tables 6 and 7, a field test was conducted in an environment characterised by low vegetation and intermittent wooded areas (see Fig. 7). This selection was deliberate to allow signal dissipation from the satellites. The test involved traversing a predetermined path to facilitate presentation of the results on a map interface.

Throughout the test, a series of screenshots were captured to provide a precise snapshot of the satellite configuration during a specific period. These screenshots offer a visual depiction of the simultaneous utilisation of satellites from various constellations to compute the position accurately.

As depicted in Fig. 8a and b, several screenshots were captured throughout the test, illustrating the real-time information regarding the status of the satellite constellation as generated by RTKLib in NMEA0183 format messages, which were instantly presented to the user and updated with each epoch. This real-time visualisation offers users an insight into the immediate state of the satellites used to derive the positioning solution. The "Skyplot" tab provides a visual representation of the satellite positions, revealing that all utilised satellites maintain an elevation angle exceeding 15 degrees, as stipulated in the configuration. Furthermore, the "Info" tab offers a comprehensive view of the C/No ratios, signifying that most satellites are transmitting signals with satisfactory power levels.

Upon concluding the test, screenshots of the "Map" tab were captured, showcasing the final positioning results. As depicted in Fig. 9a and b, the test yielded satisfactory results. The commencement and culmination points of the test are indicated by the blue dot in the southern region. This dense



(a) (b)

Fig. 6 Interface examples 3. **a** Skyplot tab displaying the GNSSStatus information. **b** Info tab displaying the GNSSStatus information

(a) (b)

Fig. 7 Data collection area. **a** Visual depiction of the designated area, illustrating the presence of wooded regions. **b** Mapped footprint of the designated route

Table 6 Hardware specifications

Device	Android	Pseudorange	ADRM	Systems	Network
Google Pixel 7 Pro	13	Yes	Yes	GPS GAL GLO BDS	5G Vodafone

Table 7 RTKLib configuration

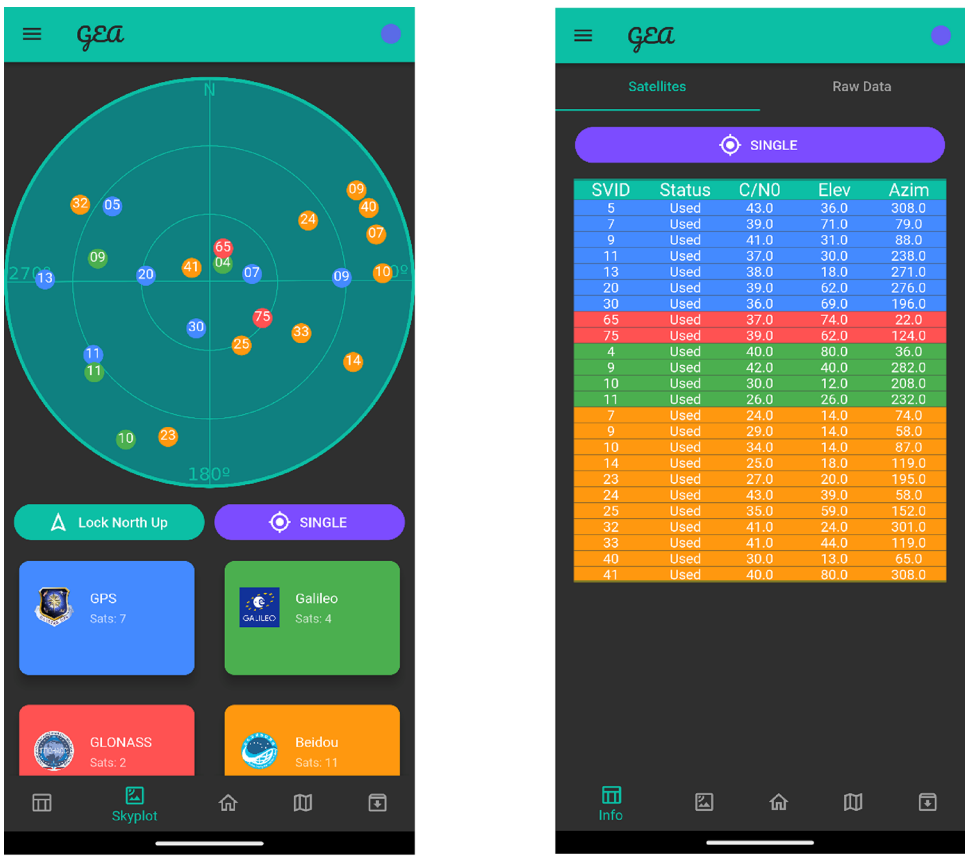
Options	Settings Value
Positioning mode	Single
Frequencies	L1
Elevation mask	15°
Ionosphere correction	Broadcast
Troposphere correction	Saastamoinen
Satellite ephemeris/Clock	Broadcast
systems	GPS, GAL, GLO, and BDS

concentration of points in that area arises from the smartphone momentarily halting during the initialisation phase when the error in the initial points is greater. After initialisation, the journey proceeds counterclockwise.

In the final section of the route, corresponding to the western zone, the positioning solution exhibited a higher error. This discrepancy could be attributed to the denser tree coverage, which adversely affects the quality of signal acquisition. A similar trend is observed in the southern region. Conversely, the eastern zone, which is characterised by a clear sky, shows a more accurate positioning solution. This segment registered lower error levels and maintained continuity without any notable outliers.

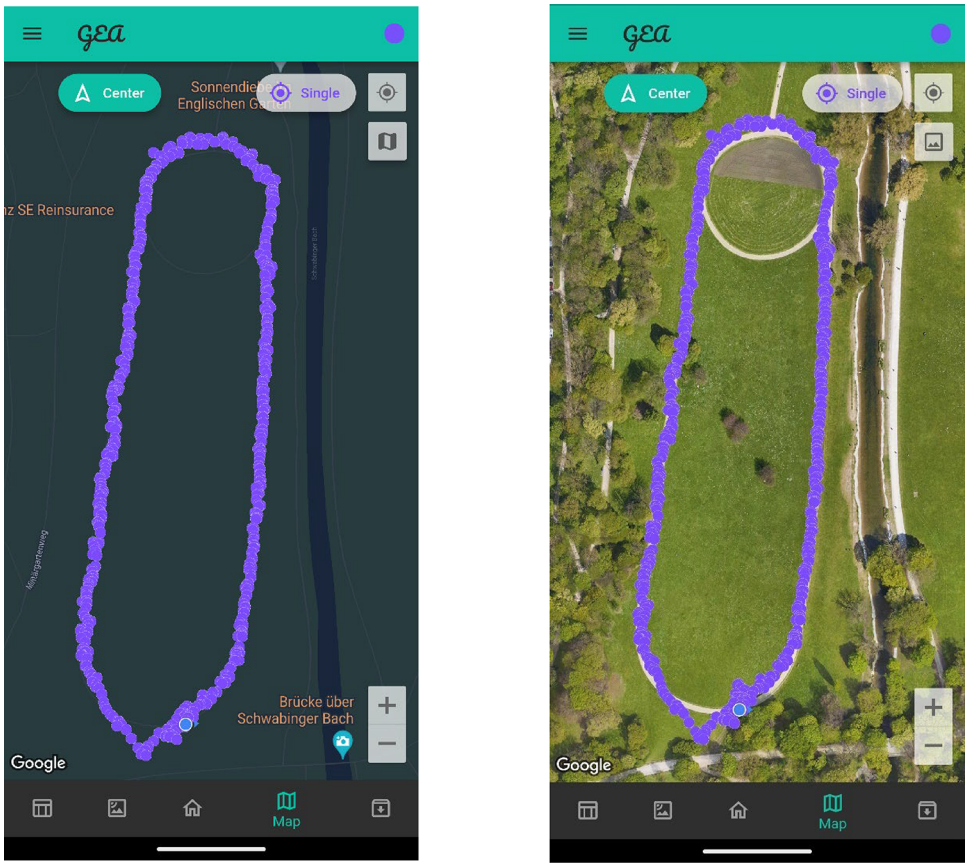
To conduct a more rigorous evaluation of the results, the app's saved raw data are employed to compute the same trajectory in a post-processing scenario. This approach aimed to establish a true reference for comparing real-time results. By doing so, it becomes possible to ascertain a relative error that accounts for any latency (which is absent in post-processing) and potential data loss during real-time processing. For post-processing of the raw data, an identical version of RTKLib was employed, along with the processing configuration detailed in Table 7. This ensured consistency and comparison with real-time results.

The initial observation that stood out was the disparity in the number of points (epochs) between each solution, with 493 for real-time and 468 for postprocessing. This discrepancy is attributed to the more stringent filters applied during postprocessing, resulting in the exclusion



(a) (b)

Fig. 8 Satellite status. **a** Skyplot tab showcasing the satellite arrangement during the test. **b** Info tab presenting a tabular overview of additional details concerning the satellites used in the positioning solution



(a) (b)

Fig. 9 Positioning solution. **a** Map tab illustrating the position overlaid on a cartographic layer. **b** Map tab displaying the position superimposed on an aerial orthophotograph. Maps generated using GEA application version 2.1.0

of epochs in which the positions could not be determined. As evidenced by the trajectory deviation, this led to larger errors in the excluded epochs.

In Fig. 10a and b, a comparative analysis of both solutions alongside a crop within a delimited region reveals discernible disparities. Table 8 presents a range of statistics detailing the discrepancies in each position component (E, N, and U), considering the initial epoch point of the real-time solution as the origin of the N frame. These statistics were computed considering epochs in which a positioning solution was obtained for both real-time and post-processing. The analysis reveals that even in the worst case, the error does not surpass 30 cm, with a standard deviation of approximately ± 5 cm in the Up component. These results fall within a reasonably acceptable range when considering the disparities between the real-time solution and post-processing, considering that both solutions exhibited variations of approximately 2 cm in the horizontal component and 5 cm in the vertical component.

Regarding the methodology outlined in "GNSS cloud computing workflow" section, all processes were confirmed to be executed without any issues. No problems were observed with the connection, as evidenced in the test where the position differences attributable to latency were minimal, with no significant spikes. In addition, no data loss was observed during the transmission to the server, and all messages arrived intact. Furthermore, the results produced by RTKLib were efficiently relayed between the server and the app, culminating in the successful generation of database records.

Conclusions

This study developed an integrated system, in which a smartphone app can seamlessly interface with the GNSS sensor of a device to gather a comprehensive array of raw GNSS data encompassing crucial measurements such as Pseudoranges, Carrier phase, and Doppler readings. This app initiates a dynamic process through a WebSocket connection with a dedicated server, thereby becoming a conduit for efficient data transmission and orchestration. The server, a central orchestrator, launches BNC software to generate satellite navigation information and RTKLib software for precise positioning computations, and establishes essential connections for data exchange. Notably, raw GNSS data are converted into the RTCM3 format, which is a bridge to subsequent processing. RTKLib then derives the essential measurements, and the

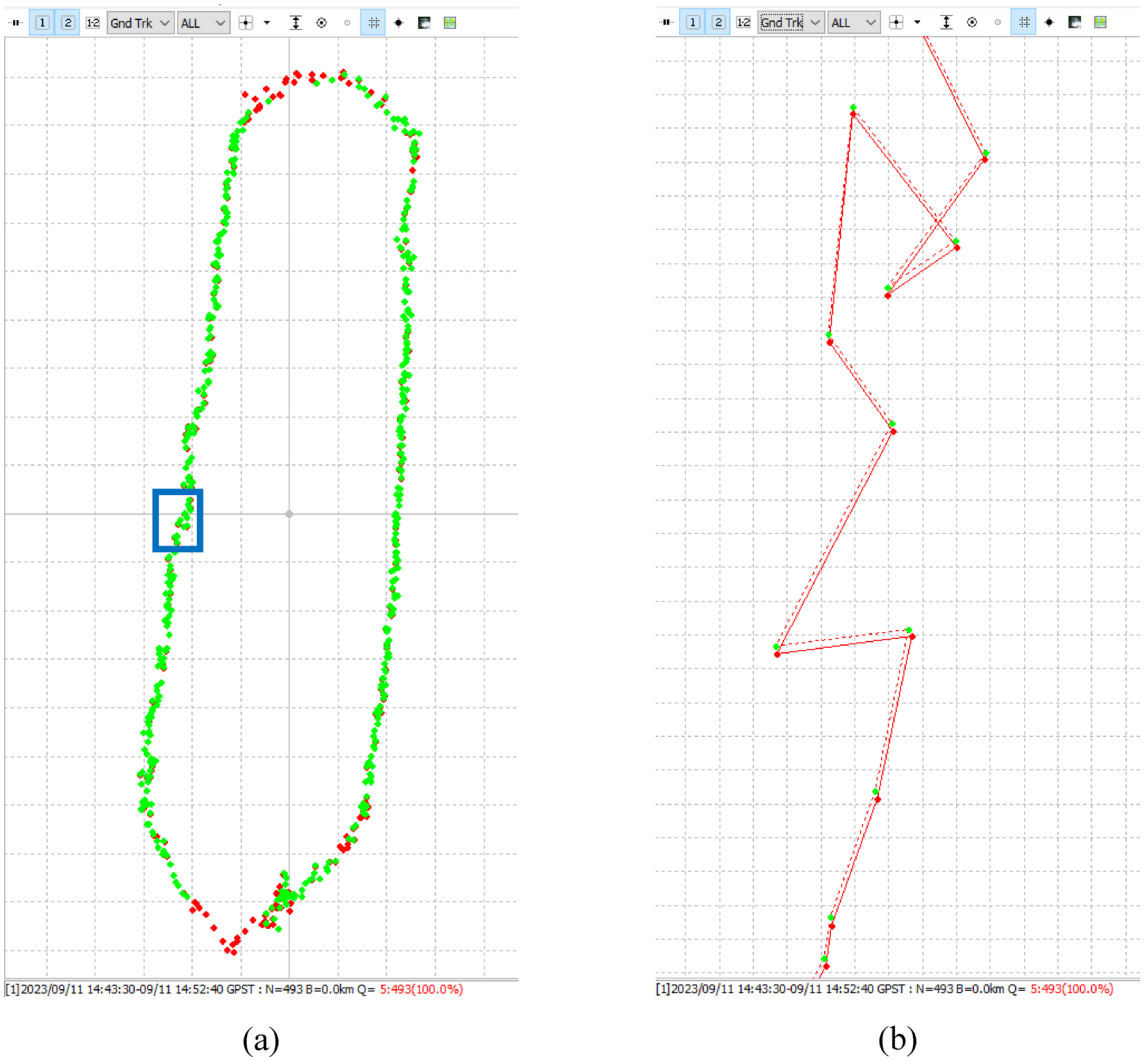


Fig. 10 Comparison of the position solution. **a** Footprint of the real-time solution (in red) juxtaposed with the post-processing solution (in green). **b** Close-up view of a specific area highlighting the disparities between the real-time and reference positions

Table 8 Statistics pertaining to the disparities between the real-time solution and post-processing results

	dE (meters)	dN (meters)	dU (meters)
Maximum (abs)	0.0025	0.0041	0.2538
Minimum (abs)	0.0000	0.0007	0.0004
Standard deviation	0.0005	0.0005	0.0418
Mean	0.0010	0.0021	0.0551
Median	0.0009	0.0021	0.0450

resulting solutions are communicated back to the app, which securely stores them. This integration ensures data integrity and reliability, which is achieved using the Firebase database. The system highlights the fusion of mobile technology, real-time communication, cloud computing, advanced processing, and cloud-based storage, highlighting its potential for advancing GNSS-based positioning applications that can encompass a range of scientific analyses, including the examination of ionospheric and tropospheric behaviours. Moreover, the scope of the app can be extended to various domains, such as urban mobility and development of smart cities.

Author contribution J.H.O. conceptualized and designed the study, outlining the framework for the Android app. J.H.O. led the programming efforts, developing the core functionalities of the app and focused on the theoretical underpinnings of the raw data conversion process. A.B.A.J. and Á.E.M.F. conducted extensive literature reviews and contributed to the methodology section. All authors have read and agreed to the published version of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Availability of data and materials The datasets generated and/or analysed during the current study are available in the GitHub repository, https://github.com/jorgeho1995/geagnss.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval Not Applicable.

References

Android for Developers (2016) *GnssMeasurement, GnssStatus, GNSSClock*. https://developer.android.com. Accessed 18 August 2023

Banville S, Van Diggelen F (2016) Precise GNSS for Everyone: Precise Positioning Using Raw GPS Measurements from Android Smartphones. GPS World 27(11):43–48

BNC (2023) BKG Ntrip Client. https://igs.bkg.bund.de/ntrip/bnc. Accessed 04 September 2023

Dutta D, Mahato S, Raja S, Ganguli S, Verma S, Bose A (2020) Android Smartphones for GNSS studies in Multi-Constellation Environment. In: National conference on emerging trends on sustainable technology and engineering applications, Durgapur, pp 7–8. https://doi.org/10.1109/NCETSTEA48365.2020.9119916

European Global Navigation Satellite System (GSA) (2016) Using GNSS Raw Measurements on Android Devices. Publications Office of the European Union, Luxembourg. https://doi.org/10.2878/449581

Favenza A, Rossi C, Pasin M, Dominici F (2014) A cloud-based approach to GNSS augmentation for navigation services. In: Proceedings of the 7th international conference on utility and cloud computing, London, UK

Firebase (2023) Firebase. https://firebase.google.com. Accessed 04 September 2023

Flutter (2023) Flutter. https://flutter.dev. Accessed 04 September 2023

García-Molina JA, Parro JM (2017) Cloud-based GNSS processing of distributed receivers of opportunity: techniques, applications and data-collection strategies. In: 6th international colloquium on scientific and fundamental aspects of GNSS/Galileo. Valencia, Spain, 25–27 October 2017

Hernández Olcina J, Anquela Julián AB, Martín Furones ÁE (2024) Navigating latency hurdles: an in-depth examination of a cloud-powered GNSS real-time positioning application on mobile devices. Sci Rep 14:14668. https://doi.org/10.1038/s41598-024-65652-7

Konstantinos E, Konstantinos N, Stathis M, Constantine P (2013) Geospatial services in the Cloud. Comput Geosci 63:116–122

Liu X, Ribot MÁ, Gusi-Amigó A, Rovira-Garcia A, Sanz J, Closas P (2021) Cloud-based single-frequency snapshot RTK positioning. Sensors 21:3688. https://doi.org/10.3390/s21113688

Lucas-Sabola V, Seco-Granados G, López-Salcedo JA, García-Molina JA, Crisci M (2016) Cloud GNSS receivers: New advanced applications made possible. In: 2016 international conference on localization and GNSS (ICL-GNSS). ENC, 2018. IEEE. https://doi.org/10.1109/ICL-GNSS.2016.7533852

Lucas-Sabola V, Seco-Granados G, López-Salcedo JA, García-Molina JA, Hein GW (2018) GNSS IoT Positioning: From Conventional Sensors to a Cloud-Based Solution. Inside GNSS. *Inside GNSS*. https://insidegnss.com/gnss-iot-positioning-from-conventional-sensors-to-a-cloud-based-solution/. Accessed 31 July 2023

Mahato S, Dutta D, Roy M, Santra A, Dan S, Bose A (2024a) Common android smartphones and apps for cost efficient gnss data collection: an overview. IETE J Res 70(2):1871–1884. https://doi.org/10.1080/03772063.2022.2164369

Mahato S, Goswami M, Kundu S, Bose A (2024b) Single baseline long distance RTK using CLS GNSS module & opensource software: case study from India. IETE J Res. https://doi.org/10.1080/03772063.2023.2192424

NodeJS (2023) NodeJS. https://nodejs.org. Accessed 04 September 2023

Radio Technical Commission for Maritime Services (RTCM) (2013) RTCM Standard 10403.2. Differential GNSS (Global Navigation Satellite System) Services – Version 3. RTCM Special Commitee No. 104, Arlington, Virginia. RTCM Paper 104-2013-SC104-STD

Robustelli U, Baiocchi V, Pugliano G (2019) Assessment of dual frequency GNSS observations from a Xiaomi Mi 8 android smartphone and positioning performance analysis. Electronics. https://doi.org/10.3390/electronics8010091

RTKLib (2013) RTKLib ver. 2.4.2 Manual. https://www.rtklib.com/prog/manual_2.4.2.pdf. Accessed 04 September 2023

WebSockets (2023) WebSockets Standard. https://websockets.spec.whatwg.org. Accessed 04 September 2023

Jorge Hernández Olcina received her double master's degree from the Universitat Politècnica de València (UPV) and the University of Applied Sciences in Karlsruhe in 2019. He is a Ph.D. student in the Department of Cartographic Engineering, Geodesy and Photogrammetry at UPV and a GNSS R&D Engineer at Trimble Services GmbH where he works in the field of GNSS.

Ana B. Anquela Julián received her Ph.D. degree in Geodesy and Cartography at Universitat Politècnica de València (UPV) in 2001 and is a senior scientist and a lecturer professor at the Department of Cartographic Engineering, Geodesy and Photogrammetry at UPV. Her main research areas are GNSS and geodetic monitoring.

Ángel E. Martín Furones received his Ph.D. degree in Geodesy and Cartography at Universitat Politècnica de València (UPV) in 2001. He is a senior scientist and a lecturer professor at the Department of Cartographic Engineering, Geodesy and Photogrammetry at UPV. His main research areas are GNSS, Physical Geodesy, Big Data, and Machine Learning applied to Geosciences.