UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Programa de Doctorado en Ingeniería Geomática
Dpto. de Ingeniería Cartográfica, Geodesia y Fotogrametría

POSICIONAMIENTO GNSS CON COMPUTACIÓN EN LA NUBE PARA TELÉFONOS INTELIGENTES: ANÁLISIS DE LATENCIA Y ESTRATEGIAS DE OPTIMIZACIÓN PARA APLICACIONES EN TIEMPO REAL

JORGE HERNÁNDEZ OLCINA

Directores:
Ana Belén Anquela Julián
Ángel Esteban Martín Furones

Octubre, 2024

"asturianos de braveza, vascos de piedra blindada,
valencianos de alegría y castellanos de alma,
labrados como la tierra y airosos como las alas;
andaluces de relámpago nacidos entre guitarras
y forjados en los yunques torrenciales de las lágrimas;
extremeños de centeno, gallegos de lluvia y calma,
catalanes de firmeza, aragoneses de casta,
murcianos de dinamita frutalmente propagada,
leoneses, navarros, dueños del hambre, el sudor y el hacha,
reyes de la minería señores de la labranza,
hombres que entre las raíces, como raíces gallardas,
vais de la vida a la muerte, vais de la nada a la nada:
yugos os quieren poner gentes de la hierba mala,
yugos que habéis de dejar rotos sobre sus espaldas."

Vientos del pueblo me llevan
Miguel Hernández, 1937

A mis queridos padres, Rosa y Lucinio, y a mi hermana Alicia,
que como los vientos del pueblo han sido fuerza y guía en mi vida.

Posicionamiento GNSS con computación en la nube para teléfonos inteligentes: análisis de latencia y estrategias de optimización para aplicaciones en tiempo real

Jorge Hernández Olcina

Resumen

El uso generalizado de teléfonos inteligentes equipados con receptores GNSS (Sistemas Globales de Navegación por Satélite) ha generado una gran cantidad de datos de posicionamiento. En este contexto, la computación en la nube en tiempo real surge como un enfoque prometedor para aprovechar esta riqueza de información de ubicación. La tesis doctoral se centra en el análisis de latencia y estrategias de optimización para este tipo de aplicaciones.

Se presenta el desarrollo de una aplicación Android que captura datos GNSS sin procesar de teléfonos inteligentes, aprovecha los recursos de la computación en la nube, calcula la posición del dispositivo y devuelve la solución computada al usuario. Esta integración no solo conserva los recursos del dispositivo, sino que también permite el cálculo de la posición en tiempo real, lo que abre el camino para aplicaciones y servicios basados en la ubicación mejorados.

En segundo lugar, la tesis analiza los desafíos de latencia dentro de las soluciones GNSS basadas en la nube. Se investiga y cuantifica el impacto de diversos factores en el sistema, incluyendo la adquisición de señales GNSS, la transmisión de datos, el procesamiento en la nube y la difusión de resultados. Se realizan experimentos controlados y escenarios del mundo real para evaluar la influencia de las condiciones de la red, las capacidades del dispositivo y la carga del servidor en la nube en la latencia general de posicionamiento. Los resultados resaltan los cuellos de botella del sistema y sus contribuciones relativas a la latencia. Adicionalmente, se presentan recomendaciones para mitigar estos desafíos y garantizar una experiencia de usuario óptima para aplicaciones de posicionamiento en tiempo real.

La tesis doctoral contribuye al avance del posicionamiento GNSS con computación en la nube para teléfonos inteligentes. Se analizan los desafíos de latencia y se presentan estrategias de optimización, además de desarrollarse una serie de herramientas para facilitar el trabajo con datos GNSS procedentes de dispositivos móviles.

Abstract

The widespread use of smartphones equipped with GNSS (Global Navigation Satellite System) receivers has generated a large amount of positioning data. In this context, real-time cloud computing emerges as a promising approach to leverage this wealth of location information. The doctoral thesis focuses on the analysis of latency and optimization strategies for such applications.

The development of an Android application is presented that captures raw GNSS data from smartphones, leverages cloud computing resources, calculates the device's position, and returns the computed solution to the user. This integration not only conserves device resources but also enables real-time position calculation, paving the way for enhanced location-based applications and services.

Second, the thesis analyses latency challenges within cloud based GNSS solutions. It investigates and quantifies the impact of various factors on the system, including GNSS signal acquisition, data transmission, cloud processing, and result dissemination. Controlled experiments and real-world scenarios are made to assess the influence of network conditions, device capabilities, and cloud server load on overall positioning latency. The findings highlight system bottlenecks and their relative contributions to latency. Additionally, recommendations are presented to mitigate these challenges and guarantee an optimal user experience for real-time positioning applications.

The doctoral thesis contributes to the advancement of GNSS positioning with cloud computing for smartphones. Latency challenges are analysed, and optimization strategies are presented, in addition to developing a series of tools to facilitate working with mobile devices GNSS data.

Resum

L'ús generalitzat de telèfons intel·ligents equipats amb receptors GNSS (Sistemes Globals de Navegació per Satèl·lit) ha generat una gran quantitat de dades de posicionament. En este context, la computació en el núvol en temps real sorgix com un enfocament prometedor per a aprofitar esta riquesa d'informació d'ubicació. La tesi doctoral se centra en l'anàlisi de latència i estratègies d'optimització per a esta mena d'aplicacions.

Es presenta el desenvolupament d'una aplicació Android que captura dades GNSS sense processar de telèfons intel·ligents, aprofita els recursos de la computació en el núvol, calcula la posició del dispositiu i retorna la solució computada a l'usuari. Esta integració no sols conserva els recursos del dispositiu, sinó que també permet el càlcul de posició en temps real, la qual cosa obri el camí per a aplicacions i servicis basats en la ubicació millorats.

En segon lloc, la tesi analitza els desafiaments de latència dins de les solucions GNSS basades en el núvol. S'investiga i quantifica l'impacte de diversos factors en el sistema, incloent-hi l'adquisició de senyals GNSS, la transmissió de dades, el processament en el núvol i la difusió de resultats. Es realitzen experiments controlats i escenaris del món real per a avaluar la influència de les condicions de la xarxa, les capacitats del dispositiu i la càrrega del servidor en el núvol en la latència general de posicionament. Les troballes ressalten els principals problemes del sistema i les seues contribucions relatives a la latència. Addicionalment, es presenten recomanacions per a mitigar estos desafiaments i garantir una experiència d'usuari òptima per a aplicacions de posicionament en temps real.

La tesi doctoral contribuïx a l'avanç del posicionament GNSS amb computació en el núvol per a telèfons intel·ligents. S'analitzen els desafiaments de latència i es presenten estratègies d'optimització, a més de desenvolupar-se una sèrie de ferramentes per a facilitar el treball amb dades GNSS procedents de dispositius mòbils.

Contenido

Agradecimientos

El camino hacia la finalización de esta tesis doctoral ha estado lleno de aprendizajes, desafíos y satisfacciones. No hubiera sido posible llegar a este punto sin el apoyo y la colaboración de diversas personas a quienes quiero expresar mi más profundo agradecimiento.

En primer lugar, quiero agradecer a mis directores de tesis, Ana y Ángel, por haberme introducido en el apasionante mundo del GNSS en smartphones. Su guía experta, paciencia y aliento han sido fundamentales para mi desarrollo como investigador y para la culminación exitosa de este proyecto. Agradezco especialmente su confianza en mí y por haberme permitido explorar nuevas ideas y enfoques en mi trabajo.

A mi familia y amigos, quiero expresarles mi más sincero amor y gratitud por su apoyo incondicional a lo largo de estos años. Gracias por su comprensión en los momentos difíciles, por sus palabras de aliento y por celebrar conmigo cada logro alcanzado. Su presencia en mi vida ha sido un pilar fundamental para mantener la motivación y el entusiasmo durante todo el proceso de investigación.

Finalmente, quiero agradecer a todas las personas que, de una u otra manera, han contribuido a mi formación personal y profesional. A mis profesores, mentores y amigos, les agradezco por sus enseñanzas, consejos y apoyo.

Tesis por compendio de artículos

Este trabajo de investigación se compone de artículos publicados en revistas indexadas o de renombre y sometidas a evaluación por pares, los cuales se compilan en esta tesis doctoral. Se presentan tres artículos en los cuales el doctorando es el autor principal y los directores aparecen como coautores. Tanto la Comisión Académica del Programa de Doctorado como los directores de la tesis autorizan la entrega de la tesis en este formato (tesis por compendio de artículos).

En este apartado, se detalla el listado de artículos que integran esta tesis doctoral:

- ***Artículo 1:*** Real-Time Cloud Computing of GNSS Measurements from Smartphones and Mobile Devices for Enhanced Positioning and Navigation
    - Autores: Jorge Hernández Olcina, Ana B. Anquela Julián, Ángel E. Martín Furones
    - Revista: GPS Solutions. Revista JCR situada la 13 de un total de 62 en la categoría de Remote Sensing **(JCR: Q1).** Factor de impacto: **4,5.**
    - DOI: 10.1007/s10291-024-01705-8
    - Año de publicación: 2024
    - Métricas: **683** accesos (07-09-2024).

- ***Artículo 2:*** Python toolbox for android GNSS raw data to RINEX conversion
    - Autores: Jorge Hernández Olcina, Ana B. Anquela Julián, Ángel E. Martín Furones
    - Revista: GPS Solutions. Revista JCR situada la 13 de un total de 62 en la categoría de Remote Sensing **(JCR: Q1).** Factor de impacto: **4,5.**
    - DOI: 10.1007/s10291-024-01631-9
    - Año de publicación: 2024
    - Métricas: **2530** accesos (07-09-2024).

- ***Artículo 3:*** Navigating Latency Hurdles: An In-Depth Examination of a Cloud-Powered GNSS Real-Time Positioning Application on Mobile Devices
    - Autores: Jorge Hernández Olcina, Ana B. Anquela Julián, Ángel E. Martín Furones
    - Revista: Scientific Reports. Revista JCR situada la 25 de un total de 134 en la categoría de Multidisciplinary Sciences **(JCR: Q1).** Factor de impacto: **3,8.**
    - DOI: 10.1038/s41598-024-65652-7
    - Año de publicación: 2024
    - Métricas: **496** accesos (07-09-2024).

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

1.1. Contextualización y motivación

El aterrizaje del Apolo 11 en la Luna no solo fue un hito de la exploración espacial, sino también un testimonio del ingenio humano en la ingeniería y la computación. Un componente crucial en la misión fue el uso del filtro de Kalman, un algoritmo matemático que permite estimar con precisión la posición y velocidad de una nave. Este filtro no solo se utilizó a bordo de la nave espacial Apolo, específicamente en el Sistema de Navegación y Guiado del Módulo de Comando (CNS) y en el Sistema de Navegación y Guiado del Módulo Lunar (LGC), sino que también desempeñó un papel vital en la base terrestre de control de la misión en Houston (O'Brien, 2010). En la nave, el filtro de Kalman era fundamental para procesar los datos de los sensores en tiempo real, ajustando las trayectorias de la nave con gran precisión. Sin embargo, es en la base de control donde podemos encontrar un antecedente directo de lo que hoy conocemos como computación en la nube. Mientras que los computadores a bordo se limitaban a procesar unos pocos datos localmente, en Houston, los controladores de misión utilizaban poderosos sistemas computacionales para procesar todos los datos telemétricos enviados desde la nave. Estos datos eran analizados en la base utilizando también el filtro de Kalman, lo que permitía realizar cálculos más complejos y precisos que luego eran retransmitidos a la nave para la corrección de su trayectoria (O'Brien, 2010).

Esta configuración puede verse como un precursor de la computación en la nube moderna: la nave espacial, con recursos limitados a bordo, dependía en gran medida de los cálculos realizados en la base terrestre, donde la capacidad computacional era mayor y los resultados podían ser enviados de vuelta a la nave en tiempo real. De manera similar, en la actualidad, los dispositivos móviles con capacidades limitadas de procesamiento y energía pueden aprovechar la computación en la nube para realizar tareas complejas, como el procesamiento de datos GNSS en tiempo real. Este paralelismo subraya cómo la evolución tecnológica ha llevado la idea de "computación remota" desde los centros de control de misiones espaciales hasta los dispositivos que llevamos en nuestros bolsillos.

La computación en la nube permite descargar tareas computacionalmente intensivas desde dispositivos móviles a servidores remotos, optimizando no solo la eficiencia en el procesamiento de datos, sino también la precisión en aplicaciones críticas como el posicionamiento en tiempo real. Al igual que el filtro de Kalman en la base de Houston, que permitía cálculos complejos fuera de la nave espacial, la computación en la nube ofrece una solución moderna para manejar las limitaciones de los dispositivos móviles en la era actual. Este enfoque no solo mejora la capacidad de procesamiento, sino que también permite la implementación de algoritmos más avanzados y precisos, como los utilizados en la mejora del rendimiento de los sistemas GNSS.

Sin embargo, uno de los desafíos con los que se encontró la misión del Apolo 11 fue la latencia, tanto en la comunicación entre la nave espacial y la Tierra como en el procesamiento de los datos a bordo. El tiempo que tardaba una señal de radio en viajar desde la nave hasta la base terrestre en Houston, aproximadamente 1.28 segundos debido a la distancia entre la Tierra y la Luna, suponía solo una parte del retraso (Mindell, 2008). Adicionalmente, había que considerar el tiempo de procesamiento de los datos tanto a bordo como en la base, así como los retrasos introducidos por el equipo de transmisión y los sistemas de comunicación de la época, lo que añadía varios segundos más de latencia total. Mindell (2008) destaca que, a pesar de contar con sistemas automatizados, el astronauta al mando tomó el control manual de la nave en los momentos finales. Esta decisión estaba relacionada en parte con la latencia y la necesidad de tomar decisiones rápidas basadas en la percepción humana inmediata.

Este reto histórico tiene paralelismos directos con el desafío de la latencia en el procesamiento de datos GNSS en tiempo real mediante la computación en la nube, abordado en esta tesis. Al igual que en el Apolo 11, donde la distancia física y la limitación de los recursos a bordo requerían el apoyo de sistemas remotos, en el contexto actual del posicionamiento GNSS, la computación en la nube ofrece una solución para superar las limitaciones del procesamiento local en dispositivos móviles. Sin embargo, la transmisión de datos entre el dispositivo y la nube introduce latencias adicionales. Estas incluyen el tiempo de transmisión de los datos GNSS crudos, el procesamiento en la nube y la devolución de los resultados al dispositivo móvil. Aunque esta latencia puede ser mayor que en un sistema completamente local, a lo largo de esta tesis se analiza cómo, mediante la optimización de los protocolos de comunicación y la infraestructura de la nube, es posible mitigar su impacto en el rendimiento del sistema.

1.2. Objetivos de la tesis

El propósito central de esta tesis es desarrollar y validar una metodología innovadora que aproveche la computación en la nube para mejorar el posicionamiento en tiempo real mediante dispositivos inteligentes, específicamente aquellos equipados con capacidades GNSS. Para lograr este propósito, se han establecido los siguientes objetivos específicos:

- **Planteamiento y desarrollo de una nueva metodología que permita el uso de la computación en la nube para posicionamiento en tiempo real mediante dispositivos inteligentes.**
    - **Descripción:** Este objetivo busca crear una infraestructura metodológica que permita la integración eficiente de datos GNSS crudos provenientes de dispositivos móviles con servicios de computación en la nube para el procesamiento en tiempo real.
    - **Relevancia:** La precisión y rapidez en el procesamiento de datos GNSS son cruciales para aplicaciones que requieren información de ubicación en tiempo real, como la navegación autónoma, los servicios de emergencia y las aplicaciones de realidad aumentada. Al delegar el procesamiento intensivo en la nube, se puede superar las limitaciones de hardware de los dispositivos móviles, mejorando así la eficiencia y la precisión del posicionamiento.
    - **Paquetes de trabajo:**
        - Implementar una aplicación móvil que capture y transmita datos GNSS crudos en tiempo real a un servidor en la nube.
        - Desarrollar un flujo de trabajo en la nube que reciba estos datos, los procese utilizando herramientas de software libre como RTKLib, y devuelva los resultados procesados al dispositivo móvil.
        - Asegurar que la metodología soporte múltiples constelaciones GNSS (GPS, Galileo, GLONASS, BeiDou) para mejorar la robustez y precisión del posicionamiento.

- **Generación de una plataforma de almacenamiento de datos y computación en la nube.**
    - **Descripción:** Este objetivo se enfoca en el desarrollo de una plataforma integral que no solo realice el procesamiento en tiempo real de los datos GNSS, sino que también almacene estos datos para análisis posteriores y permita la gestión eficiente de múltiples usuarios concurrentes.
    - **Relevancia:** Una plataforma robusta de almacenamiento y computación en la nube es esencial para manejar grandes volúmenes de datos GNSS generados por múltiples dispositivos en tiempo real. Además, facilita el acceso a datos históricos para análisis detallados, mejorando la capacidad de investigación y desarrollo en aplicaciones GNSS.
    - **Paquetes de trabajo:**
        - Utilizar servicios en la nube escalables, como Firebase, para gestionar el almacenamiento de datos y la sincronización en tiempo real.
        - Implementar mecanismos de seguridad y privacidad para proteger los datos transmitidos y almacenados.
        - Diseñar la arquitectura de la plataforma para soportar la concurrencia de múltiples usuarios, asegurando que el procesamiento y almacenamiento sean eficientes y no se degraden con el aumento de la carga.

- **Desarrollo de un conjunto de herramientas que permitan validar la metodología y realizar análisis de los resultados.**
    - **Descripción:** Este objetivo tiene como fin crear herramientas que faciliten la conversión de datos GNSS crudos al formato RINEX, así como herramientas para comparar y analizar los resultados obtenidos en tiempo real con los obtenidos mediante postproceso.
    - **Relevancia:** La validación de los resultados es fundamental para asegurar la precisión y confiabilidad de la metodología desarrollada. Las herramientas de conversión y análisis permiten evaluar el desempeño del sistema en diferentes condiciones y escenarios, identificando áreas de mejora y garantizando que los objetivos de precisión se cumplen.
    - **Paquetes de trabajo:**
        - Desarrollar o integrar una herramienta de conversión que transforme los datos GNSS crudos capturados por la aplicación móvil al formato RINEX estándar, facilitando así su análisis y postproceso.

        - Implementar un sistema de comparación que permita evaluar las diferencias entre los resultados en tiempo real y los obtenidos mediante postproceso, calculando métricas de error y precisión.
        - Crear visualizaciones y reportes automáticos que resuman los hallazgos del análisis, facilitando la interpretación de los datos y la toma de decisiones basadas en ellos.

- **Análisis de las latencias introducidas debido a la computación en la nube y aplicabilidad de la metodología.**
    - **Descripción:** Este objetivo se centra en identificar, medir y analizar las latencias presentes en todo el proceso de computación en la nube, desde la transmisión de datos desde el dispositivo móvil hasta la recepción de los resultados procesados.
    - **Relevancia:** La latencia es un factor crítico en aplicaciones de posicionamiento en tiempo real, ya que retrasos significativos pueden afectar la utilidad y precisión de los servicios proporcionados. Comprender y minimizar estas latencias es esencial para garantizar que la metodología desarrollada sea viable para aplicaciones que requieren respuestas inmediatas.
    - **Paquetes de trabajo:**
        - Medir las latencias en cada etapa del proceso, incluyendo la transmisión de datos, el procesamiento en la nube y la retransmisión de resultados al dispositivo móvil.
        - Analizar el impacto de estas latencias en la precisión y utilidad del posicionamiento en tiempo real.
        - Identificar y proponer optimizaciones en la arquitectura y flujo de datos que puedan reducir las latencias, tales como el uso de protocolos de comunicación más eficientes, optimización de algoritmos de procesamiento y selección de ubicaciones de servidores en la nube más cercanas geográficamente a los usuarios.

Cada uno de estos objetivos está diseñado para abordar aspectos clave en la integración de GNSS con la computación en la nube, asegurando que la metodología desarrollada sea robusta, precisa y aplicable en escenarios reales. La consecución de estos objetivos permitirá no solo mejorar el estado del arte en posicionamiento en tiempo real, sino también abrir nuevas posibilidades para aplicaciones avanzadas que dependan de información de ubicación precisa y en tiempo real.

1.2.1. Importancia del desarrollo integral:

- **Interoperabilidad:** Al desarrollar una metodología que integra dispositivos móviles con servicios en la nube, se promueve la interoperabilidad entre diferentes plataformas y tecnologías, facilitando la adopción de soluciones GNSS avanzadas en diversos contextos.
- **Escalabilidad:** La implementación de una plataforma en la nube que soporte múltiples usuarios y grandes volúmenes de datos asegura que la solución pueda escalar según las necesidades, adaptándose a un creciente número de aplicaciones y usuarios sin comprometer el rendimiento.
- **Innovación Tecnológica:** Este trabajo contribuye a la innovación en el campo de la geolocalización, aprovechando las capacidades de la computación en la nube para superar las limitaciones tradicionales de los dispositivos móviles y ofrecer soluciones más precisas y eficientes.

1.2.2. Metodología de trabajo

Para alcanzar estos objetivos, la tesis empleará una combinación de desarrollo de software, implementación de infraestructura en la nube, experimentación con datos GNSS reales y análisis estadístico de los resultados. Se llevarán a cabo pruebas exhaustivas en diferentes entornos y condiciones para evaluar el desempeño de la metodología desarrollada, asegurando su robustez y aplicabilidad en situaciones del mundo real.

1.3. Estado del arte

El uso de sistemas de posicionamiento global ha experimentado un avance significativo en las últimas décadas, impulsado por la evolución de los dispositivos móviles y la integración de tecnologías emergentes como la computación en la nube. A medida que los smartphones se han convertido en herramientas omnipresentes, la demanda de aplicaciones de posicionamiento en tiempo real ha crecido exponencialmente, lo que ha llevado a la comunidad científica y tecnológica a explorar nuevas metodologías para mejorar la precisión y eficiencia del posicionamiento GNSS

1.3.1. Evolución de los sistemas GNSS en dispositivos móviles

El desarrollo de chips GNSS de bajo costo ha sido fundamental para la democratización del acceso a servicios de posicionamiento en dispositivos móviles. Estos chips, integrados en smartphones y otros dispositivos portátiles, permiten a los usuarios obtener información precisa de su ubicación a nivel global. Sin embargo, las capacidades limitadas de procesamiento y consumo energético de estos dispositivos presentan desafíos importantes para aplicaciones que requieren una alta precisión y respuestas en tiempo real (Banville & Van Diggelen, 2016).

La evolución del sistema operativo Android ha jugado un papel crucial en esta transformación. Con la introducción de Android 7.0 (Nougat), Google permitió por primera vez el acceso a mediciones GNSS crudas en dispositivos móviles. Esto abrió nuevas posibilidades para aplicaciones que requieren un procesamiento más preciso de los datos GNSS, como el posicionamiento de punto preciso (PPP) y la cinemática en tiempo real (RTK). La disponibilidad de estas mediciones crudas, incluyendo pseudorango, fase portadora y desplazamiento Doppler, permitió mejorar significativamente la precisión del posicionamiento en dispositivos móviles (GSA, 2016).

La implementación de soluciones de doble frecuencia y multiconstelación en dispositivos como el Xiaomi Mi 8 marcó otro avance importante, permitiendo a los usuarios beneficiarse de un mayor número de señales GNSS y, por ende, mejorar la precisión y robustez del posicionamiento (Robustelli et al., 2019). Estas innovaciones han llevado a una mayor exploración del uso de tecnologías avanzadas para optimizar el rendimiento del GNSS en dispositivos móviles, especialmente en entornos urbanos donde las señales pueden estar obstruidas o degradadas.

1.3.2. Desafíos y limitaciones en el procesamiento GNSS

A pesar de los avances mencionados, los dispositivos móviles continúan enfrentando limitaciones significativas en cuanto a la capacidad de procesamiento y consumo de energía. La necesidad de realizar cálculos complejos para mejorar la precisión del posicionamiento en tiempo real puede agotar rápidamente los recursos de los dispositivos, afectando su rendimiento y duración de la batería. Además, la variabilidad en las condiciones de la señal, la geometría de los satélites y las interferencias ambientales, como el efecto multipath o la atenuación de la señal, representan desafíos adicionales para mantener una alta precisión en entornos reales (Lucas-Sabola et al. (2016); García-Molina & Parro (2017)).

Para abordar estas limitaciones, la computación en la nube ha emergido como una solución prometedora. Al delegar el procesamiento intensivo de datos a servidores remotos, se puede reducir la carga en los dispositivos móviles, optimizar el uso de energía y mejorar la eficiencia del procesamiento GNSS. Este enfoque permite manejar grandes volúmenes de datos y aplicar algoritmos más sofisticados que, de otro modo, serían inviables en un entorno de dispositivo móvil.

1.3.3. Integración de la computación en la nube en aplicaciones GNSS

La computación en la nube ofrece una serie de ventajas para el procesamiento GNSS, entre las que se incluyen la capacidad de escalar recursos según la demanda, el acceso a potentes infraestructuras de procesamiento y la posibilidad de implementar algoritmos avanzados que mejoren la precisión y la velocidad del posicionamiento. Estudios recientes han demostrado el potencial de la computación en la nube para mejorar la precisión y reducir la latencia en aplicaciones GNSS en tiempo real. Por ejemplo, Favenza et al. (2014) propusieron un enfoque basado en la nube para la mejora de servicios de navegación, utilizando servidores remotos para procesar datos GNSS y proporcionar correcciones en tiempo real a los usuarios.

Asimismo, la arquitectura cliente-servidor ha sido ampliamente explorada en este contexto. Konstantinos et al. (2013) propusieron un modelo en el que los dispositivos móviles actúan como clientes que capturan datos GNSS crudos y los transmiten a un servidor para su procesamiento, lo que permite calcular la posición del dispositivo con mayor precisión. Liu et al. (2021) también adoptaron un enfoque similar, centrándose en la posición cinemática en tiempo real (RTK), donde los datos se capturan en el servidor y se procesan para generar soluciones precisas basadas en una sincronización de tiempo precisa.

Este enfoque basado en la nube no solo mejora la precisión del posicionamiento, sino que también permite la integración de datos de múltiples constelaciones GNSS, lo que mejora la redundancia y la robustez de las soluciones de posicionamiento, especialmente en entornos adversos. Además, la capacidad de la nube para manejar grandes volúmenes de datos y realizar análisis en tiempo real abre nuevas posibilidades para aplicaciones avanzadas de geolocalización, como el monitoreo ambiental y la gestión de emergencias.

La presente tesis avanza sobre estos estudios previos al ofrecer una metodología optimizada que integra la computación en la nube con aplicaciones GNSS en tiempo real. A diferencia de investigaciones anteriores, que se enfocan principalmente en el aspecto técnico del procesamiento remoto, esta tesis introduce mejoras en varios aspectos clave. En primer lugar, se propone un esquema de procesamiento que no solo minimiza la latencia mediante la optimización de los protocolos de comunicación, sino que también aborda de manera directa el problema de la concurrencia, permitiendo que la plataforma sea capaz de manejar múltiples usuarios simultáneamente sin degradar el rendimiento. Esta capacidad de escalabilidad es un aporte significativo, dado que muchas aplicaciones GNSS modernas requieren que un gran número de dispositivos móviles puedan acceder al sistema de manera simultánea. Además, la tesis introduce un enfoque más completo para abordar la latencia. A través de un análisis detallado de las fuentes de retraso en la transmisión y procesamiento de datos, se proponen estrategias para mitigar su impacto sin comprometer la precisión del posicionamiento.

1.3.4. Aplicaciones y futuro de la computación en la nube para GNSS

El futuro de las aplicaciones GNSS en dispositivos móviles está intrínsecamente ligado a la capacidad de integrar de manera eficiente la computación en la nube. A medida que las aplicaciones de posicionamiento en tiempo real se vuelven más comunes, especialmente en áreas como la navegación autónoma, los servicios de emergencia y las aplicaciones basadas en la ubicación, la necesidad de soluciones precisas, rápidas y eficientes será más crítica que nunca.

El uso de la computación en la nube permite superar muchas de las limitaciones actuales, permitiendo a los dispositivos móviles beneficiarse de recursos computacionales remotos para realizar tareas intensivas en procesamiento. Esto no solo mejora la precisión y la velocidad del posicionamiento, sino que también abre la puerta a nuevas aplicaciones que requieren un alto grado de precisión y fiabilidad. A medida que la tecnología evoluciona, es probable que veamos una mayor integración de la computación en la nube en el ecosistema GNSS, impulsando el desarrollo de soluciones innovadoras que redefinan lo que es posible en el campo del posicionamiento global.

1.4. Relevancia de la investigación

La investigación presentada en esta tesis aborda un tema de creciente importancia en el campo de la geolocalización: el uso de la computación en la nube para mejorar el procesamiento y la precisión del posicionamiento GNSS en tiempo real mediante dispositivos móviles. Este tema es particularmente relevante en un contexto donde la precisión y la fiabilidad del posicionamiento juegan un papel crucial en una amplia gama de aplicaciones, desde la navegación y los servicios de emergencia hasta la agricultura de precisión, el monitoreo ambiental, y las infraestructuras inteligentes.

1. **Impacto en Aplicaciones de Tiempo Real:** La capacidad de obtener una posición precisa en tiempo real es esencial en numerosas aplicaciones críticas,

como los vehículos autónomos, los drones, y los sistemas de navegación avanzados. La investigación que se propone tiene el potencial de mejorar significativamente la precisión de estos sistemas, especialmente en entornos urbanos complejos donde la señal GNSS puede verse afectada por la obstrucción de edificios y otras estructuras. Al utilizar la computación en la nube, es posible realizar cálculos más complejos y precisos que no serían factibles en un dispositivo móvil limitado por su capacidad de procesamiento y consumo energético.

2. **Superación de Limitaciones Técnicas:** Los dispositivos móviles, aunque potentes, tienen limitaciones inherentes en cuanto a procesamiento y consumo de energía. Esta investigación aborda directamente estas limitaciones al proponer una metodología que descarga las tareas computacionalmente intensivas a la nube, lo que no solo mejora el rendimiento del dispositivo, sino que también permite la implementación de algoritmos más avanzados que requieren mayor capacidad de procesamiento. Esto es especialmente relevante en aplicaciones donde la precisión del posicionamiento no puede verse comprometida, como en los sistemas de asistencia al conductor o en los servicios de emergencia.

3. **Contribución a la Eficiencia Energética:** La eficiencia energética es una preocupación creciente en el desarrollo de aplicaciones móviles. Al trasladar el procesamiento intensivo a la nube, se reduce significativamente el consumo de energía en el dispositivo móvil, lo que puede prolongar la vida útil de la batería y permitir un uso más prolongado y efectivo de las aplicaciones de posicionamiento. Esta característica es particularmente importante en aplicaciones de largo plazo, como el monitoreo ambiental continuo o los sistemas de vigilancia remota.

4. **Innovación en Infraestructuras Inteligentes:** La investigación también tiene implicaciones para el desarrollo de infraestructuras inteligentes, donde la precisión del posicionamiento es esencial para la gestión eficiente de recursos, el control del tráfico, y la operación de servicios públicos. La integración de GNSS con la computación en la nube podría permitir la implementación de soluciones más avanzadas en ciudades inteligentes, mejorando la calidad de vida de los ciudadanos al optimizar la gestión de la infraestructura urbana.

5. **Progreso en el Campo de la Geolocalización:** Finalmente, esta investigación contribuye al avance del conocimiento en el campo de la geolocalización, proporcionando una metodología innovadora que puede servir de base para

futuros estudios y desarrollos tecnológicos. Al explorar nuevas formas de mejorar el rendimiento de los sistemas GNSS mediante la computación en la nube, la tesis no solo ofrece soluciones prácticas para los desafíos actuales, sino que también abre nuevas direcciones de investigación que pueden tener un impacto duradero en la disciplina.

1.5. Relación entre los artículos y los objetivos

La tesis doctoral se presenta en un formato de compendio de artículos, donde cada uno de los artículos publicados aborda un aspecto clave de la metodología propuesta. A continuación, se describe cómo cada artículo se relaciona con los objetivos específicos de la tesis:

- **Artículo 1: "Real-time cloud computing of GNSS measurements from smartphones and mobile devices for enhanced positioning and navigation"**
    - **Relación con los Objetivos:**
        - **Objetivos 1 y 2:** Este artículo es fundamental para el desarrollo y la implementación de la metodología central de la tesis, que busca integrar el procesamiento GNSS en tiempo real con la computación en la nube. Detalla el desarrollo de una plataforma que permite la captura de datos GNSS desde smartphones, su procesamiento en la nube, y la devolución de los resultados en tiempo real al dispositivo móvil. Además, aborda la implementación de una infraestructura de almacenamiento y computación en la nube, lo que cumple con los objetivos de crear una metodología robusta y una plataforma eficiente para el manejo de datos GNSS.
    - **Impacto en la Investigación:**
        - Al proporcionar una descripción detallada de la infraestructura y la metodología implementadas, este artículo demuestra la viabilidad de la computación en la nube para mejorar la precisión del posicionamiento en dispositivos móviles. Sirve como base para toda la investigación, mostrando cómo la integración de tecnologías avanzadas puede superar las limitaciones tradicionales del procesamiento GNSS en dispositivos móviles.

- **Artículo 2: "Python toolbox for android GNSS raw data to RINEX conversion"**
    - **Relación con los Objetivos:**

        - **Objetivo 3:** Este artículo se centra en el desarrollo de herramientas para la validación y análisis de resultados, uno de los objetivos específicos de la tesis. En particular, se presenta una herramienta que convierte datos GNSS crudos de dispositivos Android al formato RINEX, lo que permite una comparación precisa entre los resultados obtenidos en tiempo real y los resultados procedentes del postproceso. La herramienta desarrollada es esencial para evaluar la precisión de la metodología propuesta, permitiendo un análisis detallado de la efectividad del sistema en diferentes escenarios.
    - **Impacto en la Investigación:**
        - La capacidad de convertir datos a formato RINEX facilita la validación cruzada con otros métodos y estándares, lo que es fundamental para garantizar que los resultados obtenidos mediante la metodología propuesta sean precisos y fiables. Este artículo sienta las bases para la comparación y validación de los datos GNSS, asegurando que la metodología propuesta cumple con los estándares de precisión requeridos.

- **Artículo 3: "Navigating latency hurdles: an in-depth examination of a cloud-powered GNSS real-time positioning application on mobile devices"**
    - **Relación con los Objetivos:**
        - **Objetivo 4:** Este artículo aborda directamente el análisis de las latencias introducidas por la computación en la nube, un aspecto crítico para el éxito de la metodología propuesta. A través de un análisis detallado de los factores que contribuyen a la latencia en aplicaciones GNSS en tiempo real, el artículo identifica las principales fuentes de retraso y propone estrategias para mitigarlas. Esto es esencial para garantizar que la metodología no solo sea precisa, sino también lo suficientemente rápida para aplicaciones en tiempo real.
    - **Impacto en la Investigación:**
        - La comprensión y mitigación de la latencia es crucial para cualquier aplicación de posicionamiento en tiempo real. Este artículo proporciona un análisis profundo de cómo la latencia afecta al rendimiento del sistema, ofreciendo recomendaciones para optimizar la arquitectura de la nube y mejorar la eficiencia general del sistema, lo que es vital para aplicaciones donde el tiempo es un factor crítico.

La relación entre los artículos y los objetivos de la tesis demuestra cómo cada uno de estos trabajos contribuye de manera crucial al logro de los objetivos planteados. Esta estructura permite una comprensión integral de la investigación, desde el desarrollo de herramientas de validación hasta la implementación de la metodología central y el análisis de sus limitaciones, asegurando que todos los aspectos críticos del posicionamiento GNSS en tiempo real con computación en la nube sean abordados de manera efectiva.

A continuación, se presentan los capítulos 2, 3 y 4, donde se incluyen los tres artículos publicados que conforman esta tesis doctoral. Estos capítulos están estructurados de manera que cada artículo se contextualiza dentro del marco general de la investigación, destacando las contribuciones específicas que cada uno aporta a los objetivos globales del trabajo.

1.6. Organización de la bibliografía

La bibliografía empleada en esta tesis doctoral ha sido utilizada de manera transversal, tanto en los artículos publicados como en esta sección introductoria. Dado que las referencias citadas a lo largo del texto son compartidas entre los distintos capítulos y tienen relevancia tanto en el desarrollo teórico como en los análisis realizados en los artículos, se ha decidido agrupar toda la bibliografía en un único apartado, ubicado al final del documento (**Bibliografía**). Esta estructura permite evitar redundancias y facilitar al lector la consulta de todas las fuentes empleadas de manera centralizada, asegurando así un acceso coherente y ordenado a todas las referencias utilizadas a lo largo de la tesis.

Capítulo 2

2. Artículo 1: Real-Time Cloud Computing of GNSS Measurements from Smartphones and Mobile Devices for Enhanced Positioning and Navigation

Jorge Hernández Olcina, Ana B. Anquela Julián & Ángel E. Martín Furones.
GPS Solutions 28, 167 (2024).
doi:10.1007/s10291-024-01705-8

2.1. 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.

2.2. 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 & Van Diggelen, 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 & Van Diggelen (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 & Van Diggelen, 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 & Parro (2017), Lucas-Sabola et al. (2018)). 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 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. (2023) 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. (2024) 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 & Parro (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.

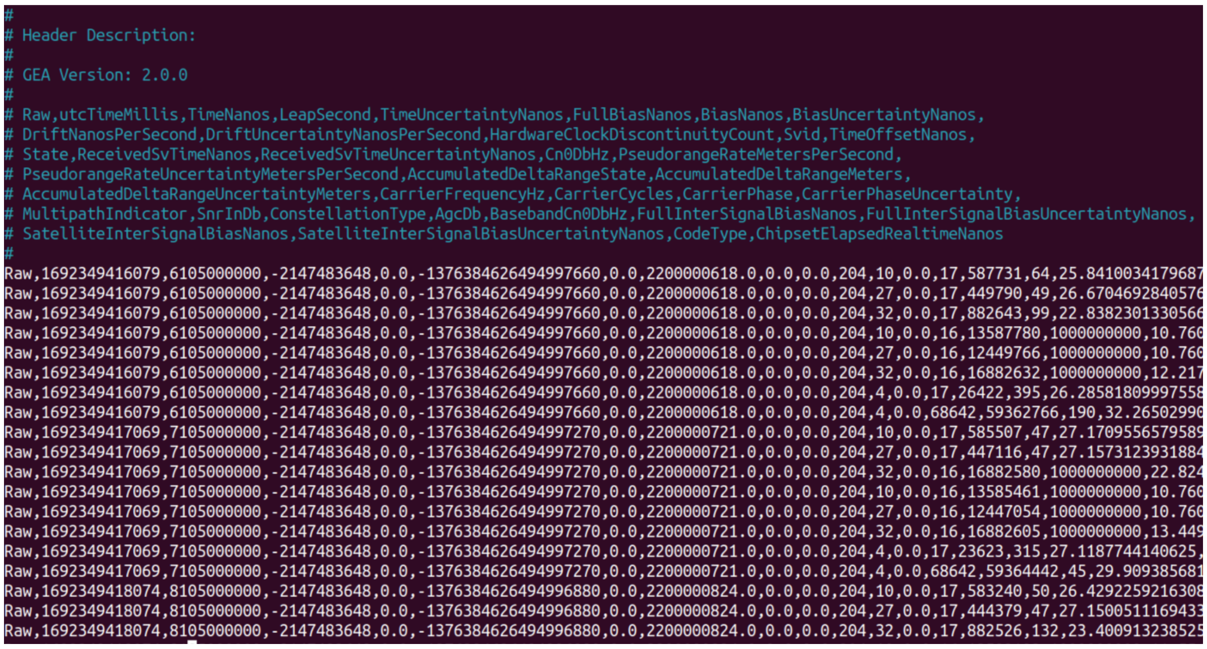
2.3. Methodology

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.

2.3.1. Front End

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 post-processing, 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 (Raw GNSS Measurements, 2016). From the raw data, a satellite message was generated, conforming to the identical structure implemented in Google's GNSSLogger app (Raw GNSS Measurements, 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.



```
#
# Header Description:
#
# GEA Version: 2.0.0
#
# Raw,utcTimeMillis,TimeNanos,LeapSecond,TimeUncertaintyNanos,FullBiasNanos,BiasNanos,BiasUncertaintyNanos,
# DriftNanosPerSecond,DriftUncertaintyNanosPerSecond,HardwareClockDiscontinuityCount,Svid,TimeOffsetNanos,
# State,ReceivedSvTimeNanos,ReceivedSvTimeUncertaintyNanos,Cn0DbHz,PseudorangeRateMetersPerSecond,
# PseudorangeRateUncertaintyMetersPerSecond,AccumulatedDeltaRangeState,AccumulatedDeltaRangeMeters,
# AccumulatedDeltaRangeUncertaintyMeters,CarrierFrequencyHz,CarrierCycles,CarrierPhase,CarrierPhaseUncertainty,
# MultipathIndicator,SnrInDb,ConstellationType,AgcDb,BasebandCn0DbHz,FullInterSignalBiasNanos,FullInterSignalBiasUncertaintyNanos,
# SatelliteInterSignalBiasNanos,SatelliteInterSignalBiasUncertaintyNanos,CodeType,ChipsetElapsedRealtimeNanos
#
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,10,0.0,17,587731,64,25.8410034179687
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,27,0.0,17,449790,49,26.6704692840576
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,32,0.0,17,882643,99,22.8382301330566
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,10,0.0,16,13587780,1000000000,10.760
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,27,0.0,16,12449766,1000000000,10.760
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,32,0.0,16,16882632,1000000000,12.217
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,4,0.0,17,26422,395,26.2858180999755
Raw,1692349416079,6105000000,-2147483648,0.0,-1376384626494997660,0.0,2200000618.0,0.0,0.0,204,4,0.0,68642,59362766,190,32.2650299
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,10,0.0,17,585507,47,27.1709556579589
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,27,0.0,17,447116,47,27.1573123931884
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,32,0.0,16,16882580,1000000000,22.824
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,10,0.0,16,13585461,1000000000,10.760
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,27,0.0,16,12447054,1000000000,10.760
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,32,0.0,16,16882605,1000000000,13.449
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,4,0.0,17,23623,315,27.1187744140625,
Raw,1692349417069,7105000000,-2147483648,0.0,-1376384626494997270,0.0,2200000721.0,0.0,0.0,204,4,0.0,68642,59364442,45,29.909385681
Raw,1692349418074,8105000000,-2147483648,0.0,-1376384626494996880,0.0,2200000824.0,0.0,0.0,204,10,0.0,17,583240,50,26.4292259216308
Raw,1692349418074,8105000000,-2147483648,0.0,-1376384626494996880,0.0,2200000824.0,0.0,0.0,204,27,0.0,17,444379,47,27.1500511169433
Raw,1692349418074,8105000000,-2147483648,0.0,-1376384626494996880,0.0,2200000824.0,0.0,0.0,204,32,0.0,17,882526,132,23.400913238525
```

Figura 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.

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.

2.3.2. 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 Section 2.6 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.

2.3.3. WebSocket Connection

The communication linkage between the app and server 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, 2024).

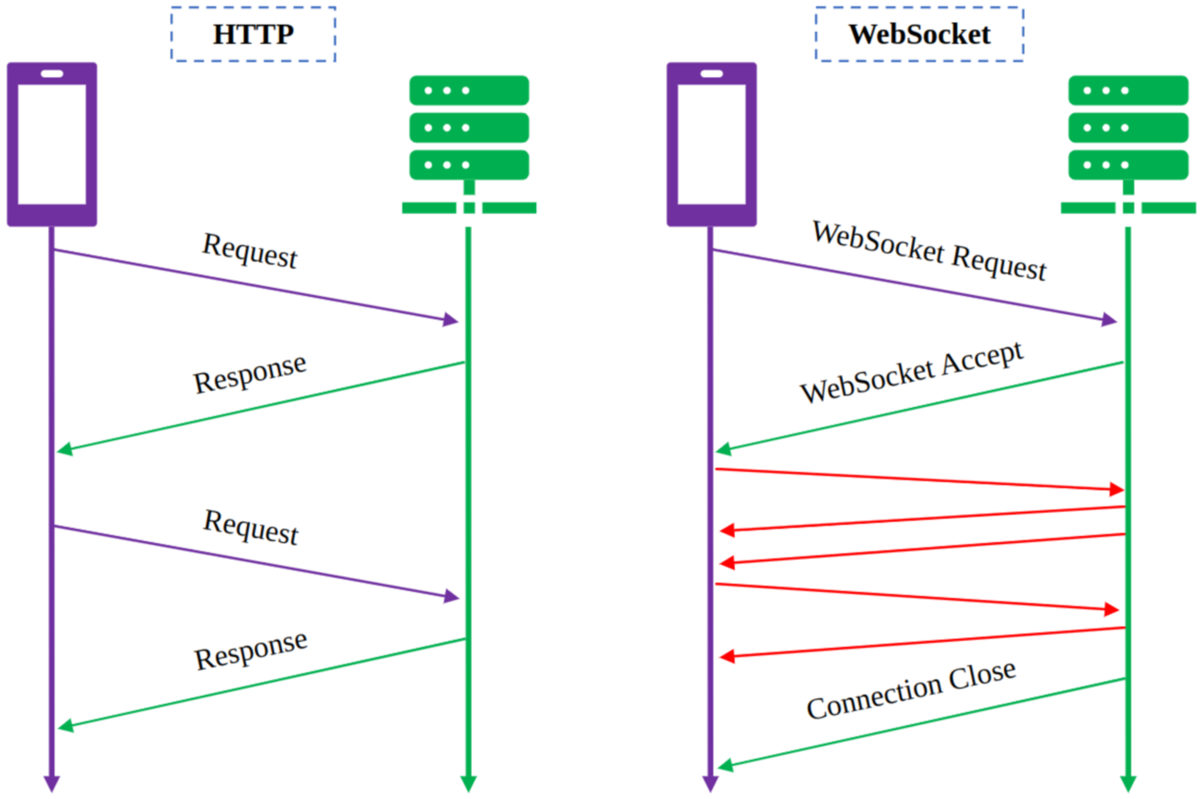


Figura 2: Comparative Diagram: HTTP vs. WebSocket Connection. As depicted in the diagram, WebSocket establishes a connection wherein both parties can exchange messages independently after initialization, eliminating the need to await the other party's response.

Here are some reasons why WebSockets are the preferred choice for real-time communication (WebSockets, 2024):

- 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 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.

2.3.4. GNSS Cloud Computing Workflow

In this integrated system, 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 Section 2.3.3. 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 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.

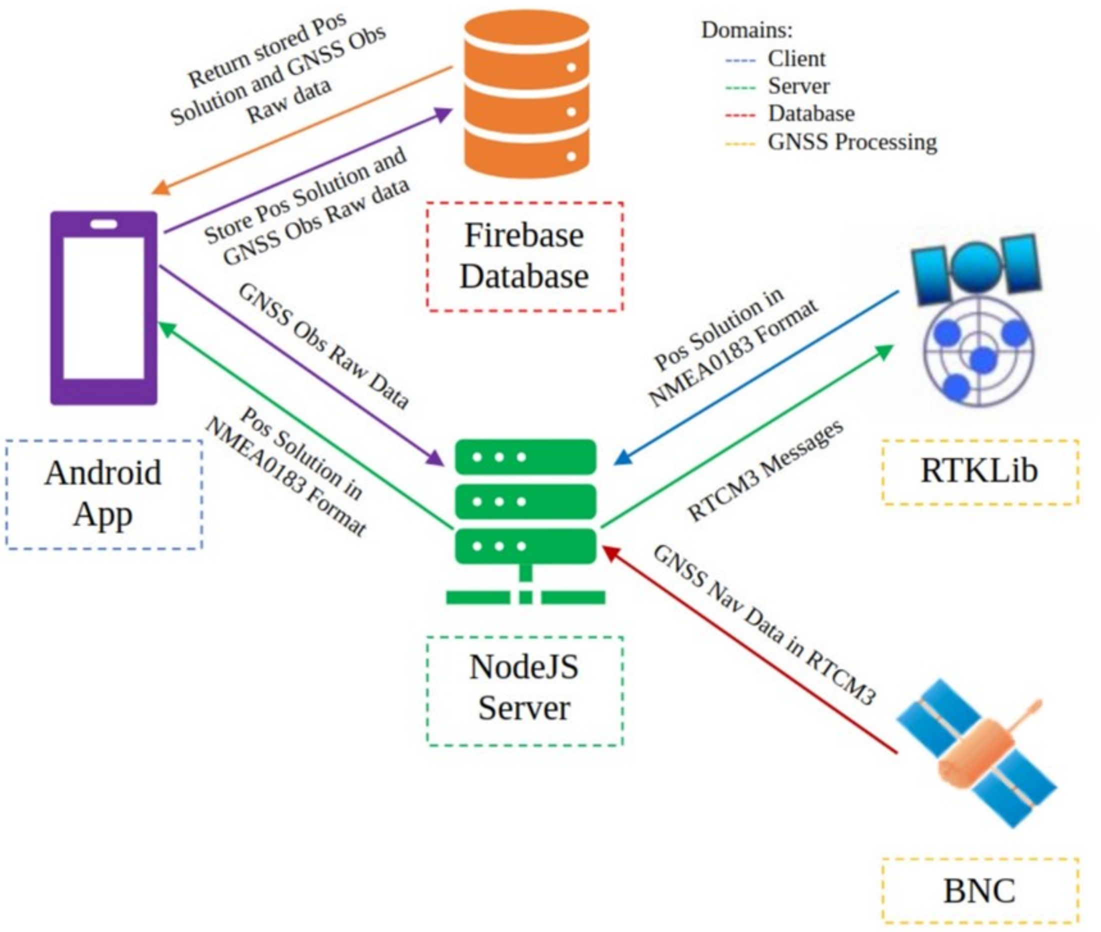


Figura 3: Workflow within the Cloud Computing platform is characterized by five pivotal components: (i) Android app, serving as a global navigation satellite system (GNSS) receiver, (ii) NodeJS server, overseeing real-time executions, (iii) BKG NTRIP Client (BNC), responsible for navigation message capture,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

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 (Raw GNSS Measurements (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:

$$GnssTime\ =\ TimeNanos - (FullBiasNanos + BiasNanos)[ns] \tag{1}$$

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:

$$GpsTime = TimeNanos - (FullBiasNanos + BiasNanos) - InterSystemsBias \tag{2}$$

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.

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

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

$$t_{Tx} = ReceivedSvTimeNanos[ns] \tag{4}$$

where *ReceivedSvTimeNanos* denotes the emission time in ns. The acceptable scope of t_{Rx} can be reconstructed in the form:

$$t_{Rx_{GNSS}} = GnssTime + TimeOffsetNanos \tag{5}$$

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.

Tabla 1: *AccumulatedDeltaRangeState* (Source Raw GNSS Measurements (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.

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 *Tabla 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 "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. *Tabla 2*, *Tabla 3* and *Tabla 4* present the predominant MSM5 message fields along with their corresponding values or formulas extracted from the raw GNSS data.

Tabla 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

Tabla 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 millisecond	$Round\left(\frac{\rho \cdot 2^{10}}{c \cdot 10^{-3}}\right) \& 1023$	(8)
GNSS satellite rough PhaseRangeRates	$Round\left(\frac{doppler \cdot c}{CarrierFreqHz}\right)$	(9)

Tabla 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)

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 *Tabla 5*.

Tabla 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.

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 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.

2.3.5. 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., 2024a).

2.3.6. 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 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.

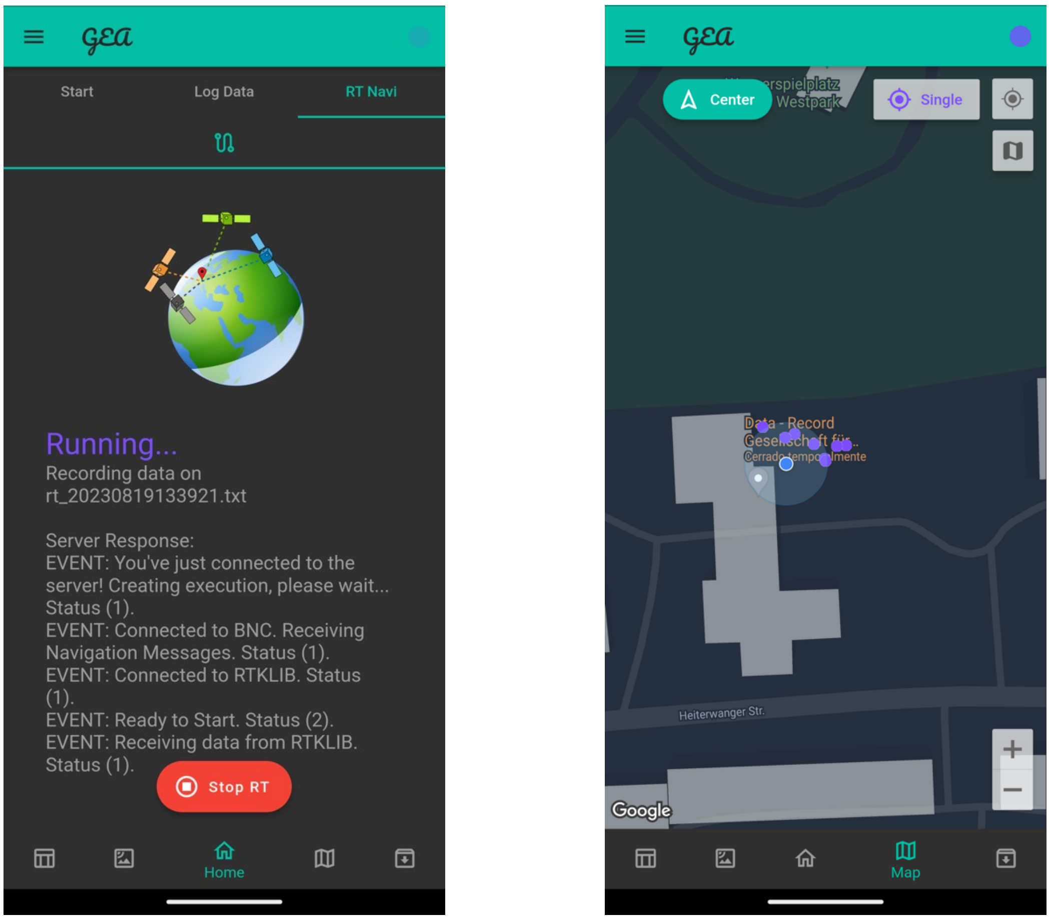
2.4. Results

This section presents the results of the implementation detailed in Section 2.3. 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.

2.4.1. App Interface

Figura 4 and *Figura 6* 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 "*Skyplot*" and "Info" tabs transition to highlighting satellite data sourced from the *GnssStatus* interface of the Android API (Raw GNSS Measurements, 2016). This interface conveys the satellite elevation and azimuth details, along with their C/No values. As illustrated in *Figura 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.



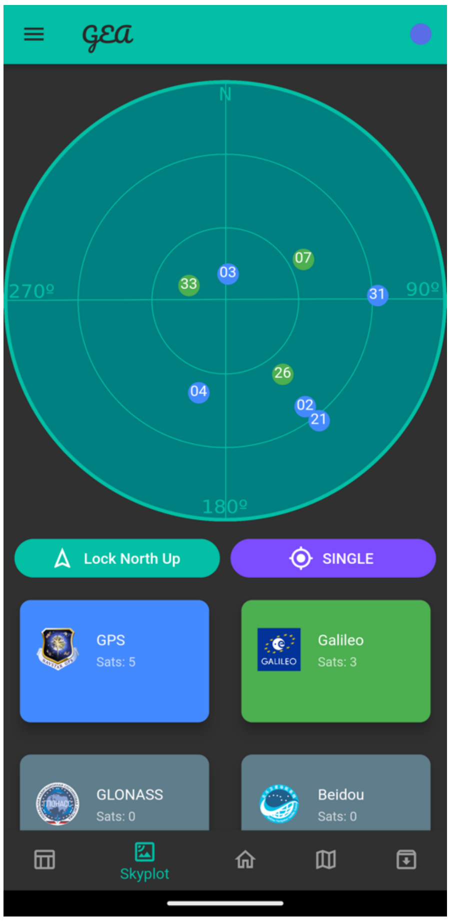
Running...
Recording data on
rt_20230819133921.txt

Server Response:
EVENT: You've just connected to the server! Creating execution, please wait... Status (1).
EVENT: Connected to BNC. Receiving Navigation Messages. Status (1).
EVENT: Connected to RTKLIB. Status (1).
EVENT: Ready to Start. Status (2).
EVENT: Receiving data from RTKLIB. Status (1).

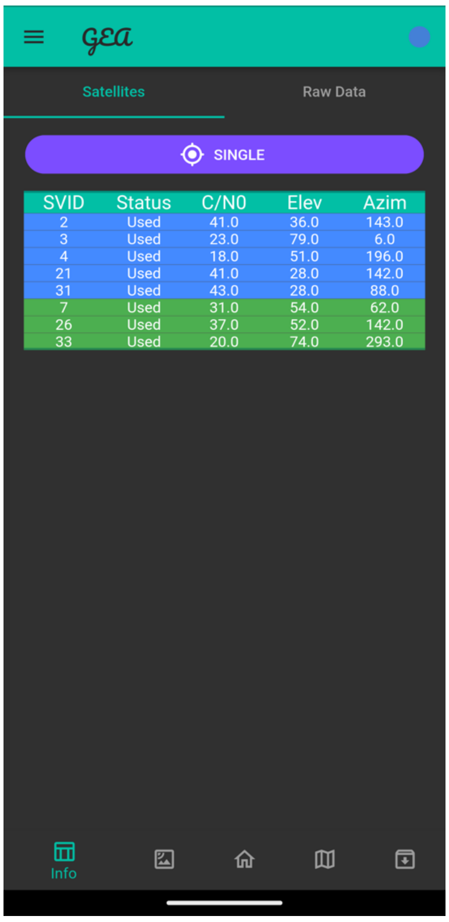
(a) (b)

Figura 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.

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



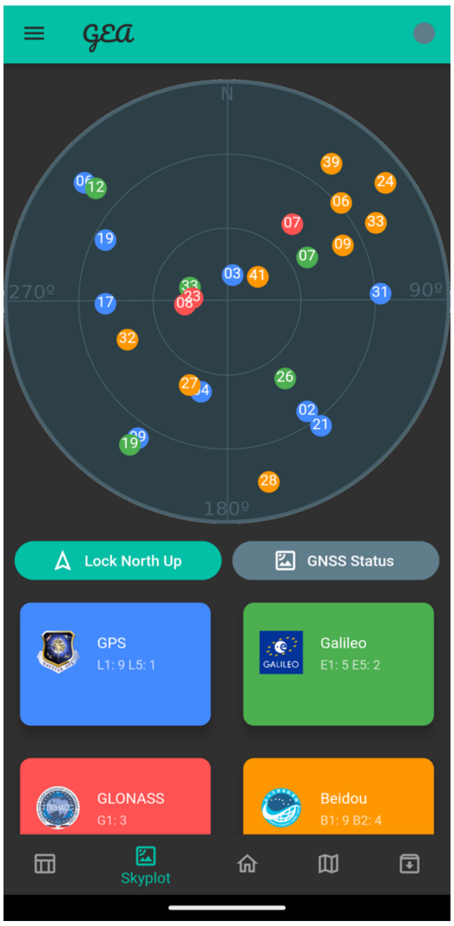
(a)



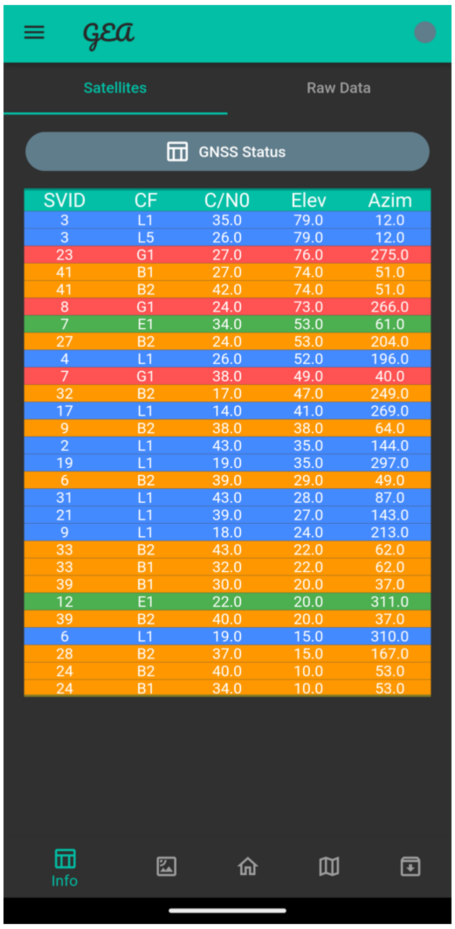
SVID	Status	C/N0	Elev	Azim
2	Used	41.0	36.0	143.0
3	Used	23.0	79.0	6.0
4	Used	18.0	51.0	196.0
21	Used	41.0	28.0	142.0
31	Used	43.0	28.0	88.0
7	Used	31.0	54.0	62.0
26	Used	37.0	52.0	142.0
33	Used	20.0	74.0	293.0

(b)

Figura 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.



(a)



SVID	CF	C/N0	Elev	Azim
3	L1	35.0	79.0	12.0
3	L5	26.0	79.0	12.0
23	G1	27.0	76.0	275.0
41	B1	27.0	74.0	51.0
41	B2	42.0	74.0	51.0
8	G1	24.0	73.0	266.0
7	E1	34.0	53.0	61.0
27	B2	24.0	53.0	204.0
4	L1	26.0	52.0	196.0
7	G1	38.0	49.0	40.0
32	B2	17.0	47.0	249.0
17	L1	14.0	41.0	269.0
9	B2	38.0	38.0	64.0
2	L1	43.0	35.0	144.0
19	L1	19.0	35.0	297.0
6	B2	39.0	29.0	49.0
31	L1	43.0	28.0	87.0
21	L1	39.0	27.0	143.0
9	L1	18.0	24.0	213.0
33	B2	43.0	22.0	62.0
33	B1	32.0	22.0	62.0
39	B1	30.0	20.0	37.0
12	E1	22.0	20.0	311.0
39	B2	40.0	20.0	37.0
6	L1	19.0	15.0	310.0
28	B2	37.0	15.0	167.0
24	B2	40.0	10.0	53.0
24	B1	34.0	10.0	53.0

(b)

Figura 6: Interface examples 3. (a) Skyplot tab displaying the GNSSStatus information. (b) Info tab displaying the GNSSStatus information.

2.4.2. Experiment Setup and Results

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

Tabla 6: Hardware specifications.

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

Tabla 7: RTKLib configuration.

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

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 *Tabla 6* and *Tabla 7*, a field test was conducted in an environment characterised by low vegetation and intermittent wooded areas. 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.

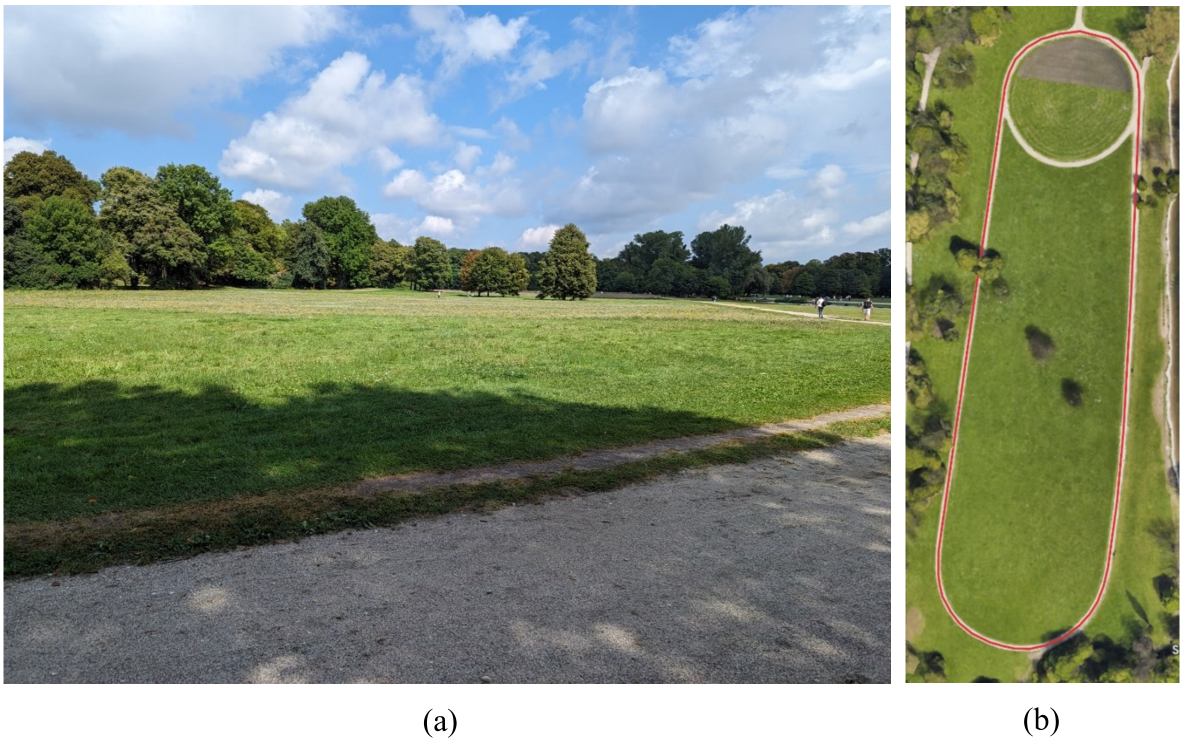
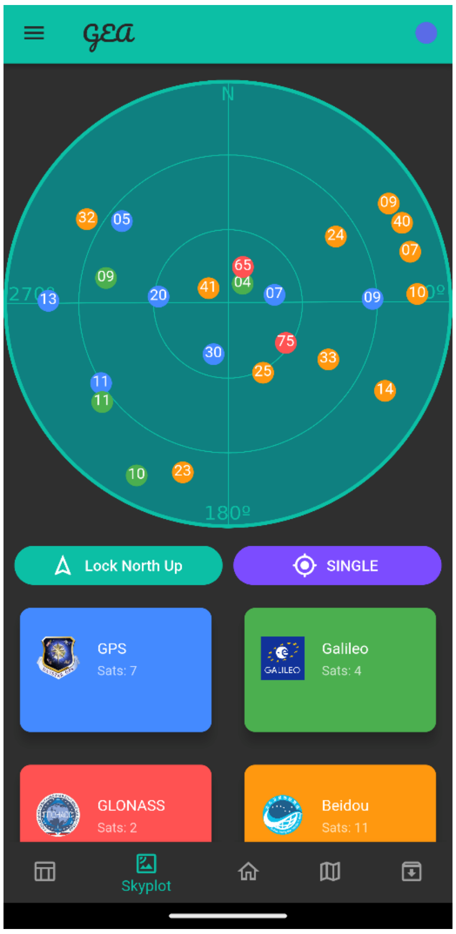
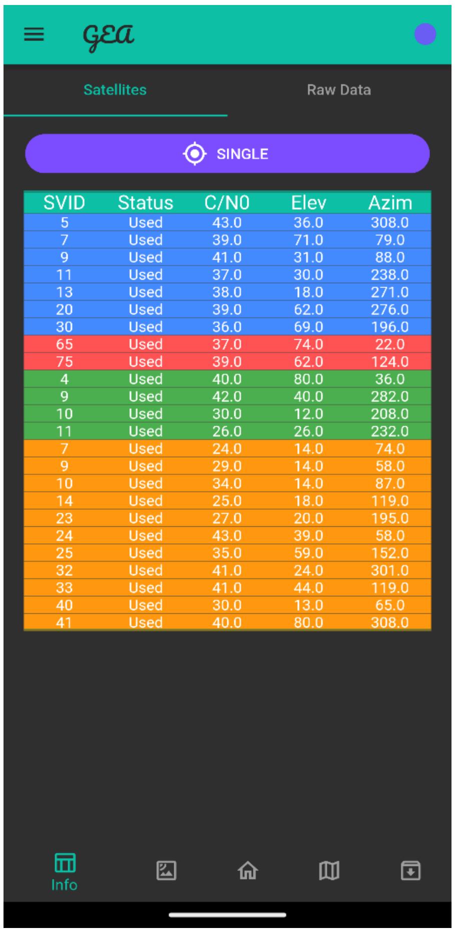

(a) (b)

Figura 7: Data collection area. (a) Visual depiction of the designated area, illustrating the presence of wooded regions. (b) Mapped footprint of the designated route.

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.



(a)



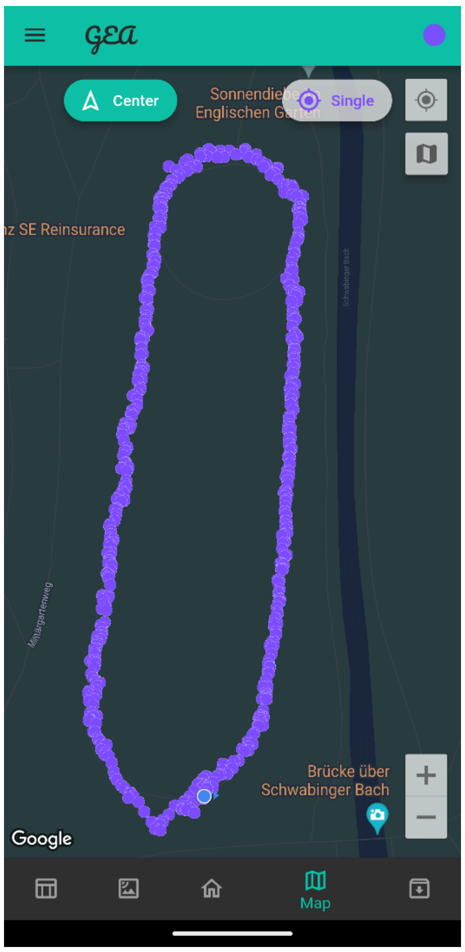
SVID	Status	C/N0	Elev	Azim
5	Used	43.0	36.0	308.0
7	Used	39.0	71.0	79.0
9	Used	41.0	31.0	88.0
11	Used	37.0	30.0	238.0
13	Used	38.0	18.0	271.0
20	Used	39.0	62.0	276.0
30	Used	36.0	69.0	196.0
65	Used	37.0	74.0	22.0
75	Used	39.0	62.0	124.0
4	Used	40.0	80.0	36.0
9	Used	42.0	40.0	282.0
10	Used	30.0	12.0	208.0
11	Used	26.0	26.0	232.0
7	Used	24.0	14.0	74.0
9	Used	29.0	14.0	58.0
10	Used	34.0	14.0	87.0
14	Used	25.0	18.0	119.0
23	Used	27.0	20.0	195.0
24	Used	43.0	39.0	58.0
25	Used	35.0	59.0	152.0
32	Used	41.0	24.0	301.0
33	Used	41.0	44.0	119.0
40	Used	30.0	13.0	65.0
41	Used	40.0	80.0	308.0

(b)

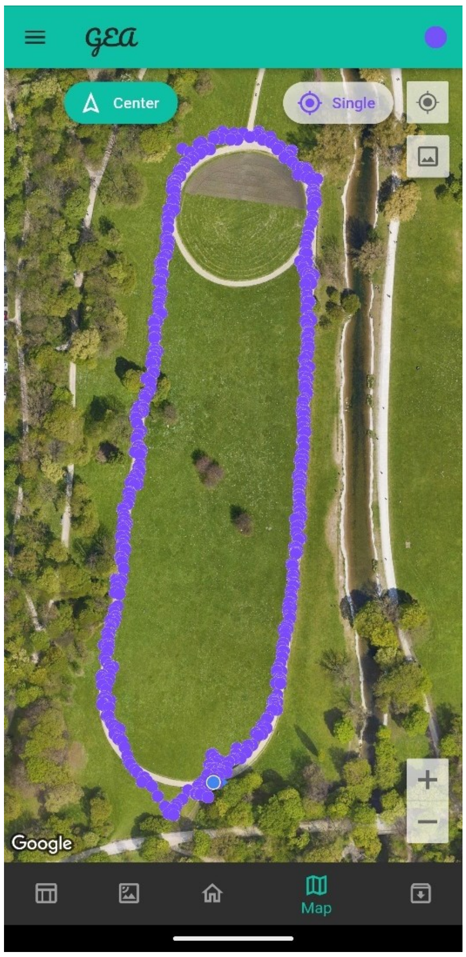
Figura 8: Satellite status. (a) Skyplot tab showcasing the satellite arrangement during the test. (b) Info tab presenting a tabular overview of additional details concerning the satellites used in the positioning solution.

As depicted in *Figura 8*, 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 *Figura 9*, 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 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.



(a)

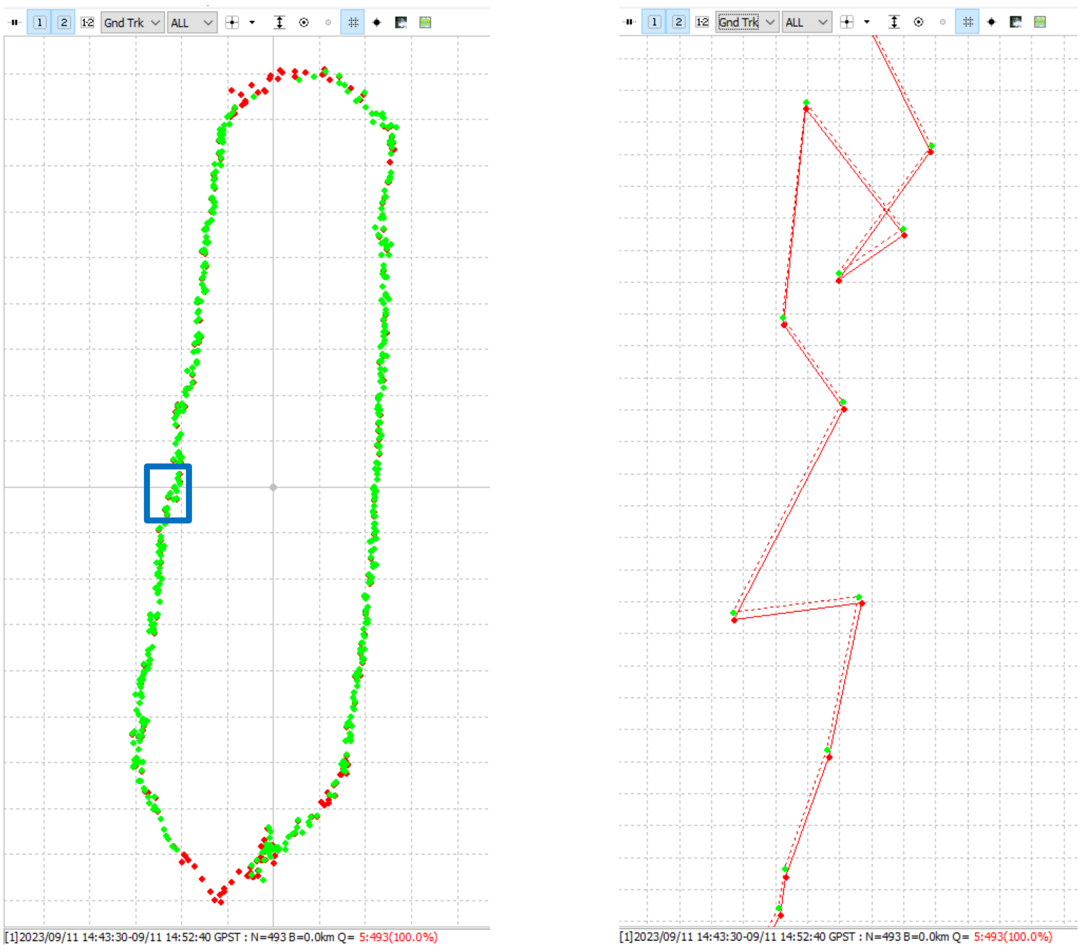


(b)

Figura 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.

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 *Tabla 7*. This ensured consistency and comparison with real-time results.



[1]2023/09/11 14:43:30-09/11 14:52:40 GPST : N=493 B=0.0km Q= 5:493(100.0%)

[1]2023/09/11 14:43:30-09/11 14:52:40 GPST : N=493 B=0.0km Q= 5:493(100.0%)

(a) (b)

Figura 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

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 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.

Tabla 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

In *Figura 10*, a comparative analysis of both solutions alongside a crop within a delimited region reveals discernible disparities. *Tabla 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 centimetres, with a standard deviation of approximately ±5 centimetres 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 Section 2.3.4, 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.

2.5. 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 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.

2.6. Data availability

The datasets generated and/or analysed during the current study are available in the GitHub repository. (*jorgeho1995*)

Capítulo 3

3. Artículo 2: Python toolbox for android GNSS raw data to RINEX conversion

Jorge Hernández Olcina, Ana B. Anquela Julián & Ángel E. Martín Furones.
GPS Solutions 28, 95 (2024).
doi:10.1007/s10291-024-01631-9

3.1. Abstract

Global navigation satellite system (GNSS) data collected from Android devices have gained increasing importance in various applications, ranging from geospatial positioning to environmental monitoring. However, the lack of standardized tools for converting Android GNSS raw data into receiver independent exchange (RINEX) format poses a significant challenge for researchers and practitioners. In response to this need, we present a comprehensive Python toolbox designed to streamline the conversion process and enhance the usability of Android GNSS data. The proposed toolbox leverages Python's versatility to provide a user-friendly interface for converting Android GNSS raw data into the widely adopted RINEX format. Key features include robust data parsing algorithms, support for multiple GNSS constellations, and compatibility with diverse Android device configurations. Furthermore, the toolbox's open-source nature encourages community collaboration and allows for continual improvement and adaptation to emerging GNSS technologies. We anticipate that this Python toolbox will serve as a valuable resource for researchers and practitioners working with Android GNSS data, facilitating standardized data interchange and promoting reproducibility in GNSS-based studies.

3.2. Introduction

The advent of smartphones featuring built-in GNSS receivers has revolutionized the field of positioning technology, granting users the ability to access accurate location data on a global scale. The development of cost-effective GNSS chips has sparked a significant revolution in the realm of positioning and navigation devices. The widespread availability of GNSS-enabled smartphones has opened up new and exciting opportunities across various domains, including but not limited to location-based services, transportation management, disaster response, and environmental monitoring. (Realini et al., 2017).

At the "I/O 2016" conference held in May 2016, Google made a notable announcement regarding the release of raw GNSS measurements from Android devices running the Android N operating system, also known as "Nougat," version 7 (GSA, 2016). Following this, in May 2018, the market saw the debut of the Xiaomi MI 8, the first mobile phone supporting multiple satellite constellations and dual frequencies (Robustelli et al., 2019). These developments mark significant milestones, ushering in a new era in positioning and navigation. The implications of these advancements are profound for the scientific community, signaling a departure from the traditional GNSS receiver black box model, which previously provided accuracy at the metric level. With access to pseudorange, Doppler, and carrier phase measurements, the door has opened for achieving more precise positions (Robustelli et al., 2019). These breakthroughs could transform the field of positioning and navigation, offering fresh possibilities for improved accuracy and performance.

Zangenehnejad et al. (2023) offer a comprehensive examination of contemporary smartphone positioning research in their work. They delve into recent breakthroughs, existing challenges, and prospective outlooks in the field. The discussion encompasses the availability of raw GNSS measurements, especially with the advent of Android 7 and the creation of Android GNSS loggers. The paper underscores the significance of scrutinizing the quality of smartphone GNSS observations for achieving high-precision positioning. Furthermore, it sheds light on the hurdles associated with smartphone GNSS positioning, encompassing factors like observation quality, device variations, environmental influences, and algorithmic advancements.

With the ubiquity of smartphones and the increasing accuracy of GNSS receivers embedded in these devices, the demand for utilizing GNSS data from Android platforms in research and applications has grown. However, the proprietary nature of the raw GNSS data collected by Android devices often poses a challenge for researchers and developers seeking to integrate this data into standard GNSS processing workflows. Hence, there is a necessity to develop a tool facilitating the conversion of raw GNSS data obtained from smartphones into a standardized format, such as RINEX.

RINEX is widely recognized as the standard for storing GNSS data, enabling the seamless exchange of raw satellite measurements across various GNSS processing software platforms. RINEX is a data interchange format for raw satellite navigation system data used in the field of geodesy. It allows the storage of measurements of pseudorange, carrier phase, Doppler, and signal-to-noise from various satellite navigation systems, including GPS, GLONASS, Galileo, BeiDou, SBAS, QZSS, and IRNSS. RINEX files are the industry standard for GNSS data, enabling software-independent processing and sequential file combination (Romero I, 2020).

The creation of tools like the Rokubun (2020) Android GNSS Logger to RINEX Converter has played a crucial role in various research endeavors. For instance, Everett et al. (2022) utilized this Python script to convert data from Google's GNSS measurement tools to RINEX. Their research focused on optimizing the application of RTKLIB for measurements conducted with Android smartphones. They emphasized the necessity for adjustments due to the suboptimal quality of measurements from mobile platforms.

However, recent updates to the Google Application Programming Interface (API) for accessing raw data and enhancements to the GNSSLogger app (Raw GNSS Measurements, 2016) have prompted the development of new conversion tools. These tools aim to accommodate the updated input data while maintaining backward compatibility with previous versions.

Hence, this article introduces *raw_android_to_rinex*, a Python toolbox designed to bridge this gap by facilitating the conversion of Android GNSS raw data to the widely used RINEX format, enabling seamless integration with existing GNSS processing tools. This tool would enhance interoperability within the GNSS community, ensuring efficient data sharing and utilization across different applications and systems.

3.3. Features of *raw_android_to_rinex*

In this section, we highlight key features of *raw_android_to_rinex*: compatibility with diverse Android devices, flexibility in handling various GNSS constellations, user-friendly interface for easy initiation, and its open-source nature, fostering collaboration and customization for users.

- Compatibility: *raw_android_to_rine*x supports raw GNSS data collected from a variety of Android devices, ensuring compatibility with different manufacturers and models.

- Flexibility: The toolbox provides flexibility in handling different GNSS constellations, including GPS, GLONASS, Galileo, BeiDou, SBAS, QZSS, and IRNSS. This flexibility ensures that researchers can work with data from a wide range of satellites.
- Ease of Use: *raw_android_to_rinex* is designed with a user-friendly interface, making it accessible to both novice and experienced users. The conversion process can be initiated with a simple command, and the toolbox includes comprehensive documentation for guidance.
- Open Source: *raw_android_to_rinex* is an open-source project, allowing users to contribute to its development, report issues, and customize the toolbox according to their specific needs.

3.4. Workflow

Having elucidated the comprehensive workflow of *raw_android_to_rinex*, it is imperative to underscore the seamless progression from data input through processing and RINEX conversion. The subsequent sections expound upon each pivotal stage, as depicted in *Figura 11*, elucidating the intricacies involved in transforming raw GNSS data from Android devices into standardized RINEX files. This elucidation aims to provide a thorough understanding of the methodology employed, ensuring clarity for users and facilitating the optimal utilization of the designated tool.

3.4.1. Data Input

To transform raw GNSS data from an Android device, the initial step involves capturing the data using an app equipped for this purpose. Presently, the tool accommodates data inputs from the logs of both the GEA and GNSSLogger apps. These apps save the raw data in a uniform format, aligning with the structure proposed by the GNSSLogger app. This standardized data input format ensures seamless compatibility, simplifying the conversion process and streamlining the utilization of raw GNSS data within the designated tool. For more details, please refer to Sect. 4.3 in the attached user manual of the tool.

The GEA app, developed by the authors of this article, serves as a specialized tool for processing raw GNSS data extracted from Android devices. This application offers dual functionalities, enabling the storage of raw GNSS data for subsequent post-processing and facilitating real-time positioning calculations through cloud computing. By providing users with the capability to both archive data for later analysis and perform on-the-fly positioning computations, the GEA app stands as a versatile solution for managing and utilizing GNSS information effectively. In this context, the noteworthy functionality is the ability to store raw GNSS data in plain text files, utilizing a format consistent with that of the GNSSLogger app. The app is available from GitHub website at *jorgeho1995*.

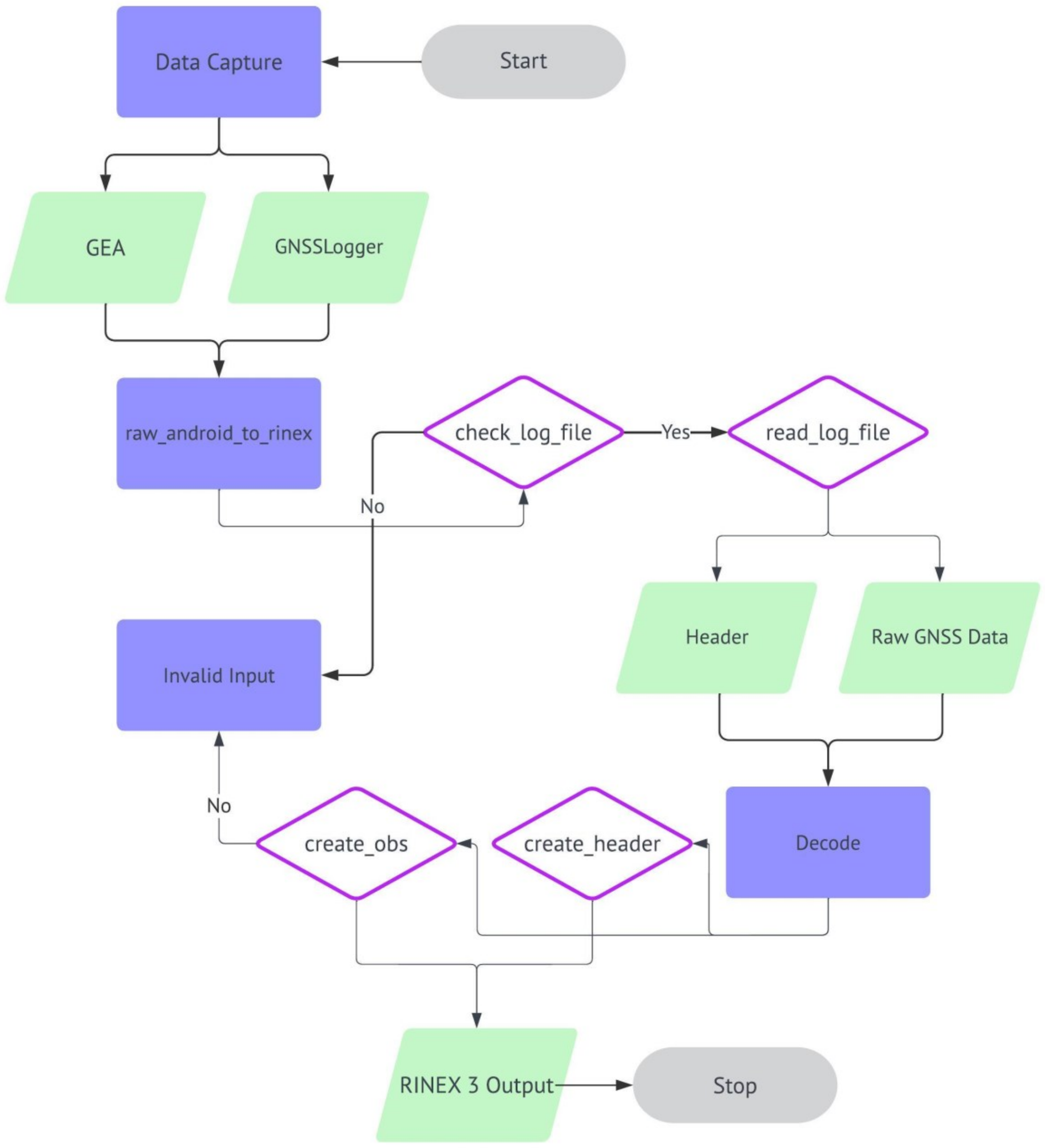


Figura 11: Overall architecture of *raw_android_to_rinex*

The GEA app, developed by the authors of this article, serves as a specialized tool for processing raw GNSS data extracted from Android devices. This application offers dual functionalities, enabling the storage of raw GNSS data for subsequent post-processing and facilitating real-time positioning calculations through cloud computing. By providing users with the capability to both archive data for later analysis and perform on-the-fly positioning computations, the GEA app stands as a versatile solution for managing and utilizing GNSS information effectively. In this context, the noteworthy functionality is the ability to store raw GNSS data in plain text files, utilizing a format consistent with that of the GNSSLogger app. The app is available from GitHub website at *jorgeho1995*.

The GNSSLogger app, created by Google, serves as a comprehensive tool for examining, visualizing, and recording location and sensor data on Android devices. This application facilitates in-depth analysis and logging of diverse data types. Notable features of the app include the ability to control data logging, view both location and raw measurement data, visualize locations on Google Maps, and analyze GNSS satellites. The app's capability extends to capturing raw GNSS measurements and other location data, systematically logging them into a file for further reference or analysis (Raw GNSS Measurements, 2016). The app is available from *Google Play*.

3.4.2. Data Processing

To obtain the pseudorange, carrier phase, Doppler measurements, and temporal values, a series of computations must be performed. As explained by the GSA (2016) since these measurements are not directly accessible through the Android API, a specific methodology is needed to determine them (Raw GNSS Measurements, 2016). For more details, please refer to Appendix A in the attached user manual of the tool. This section contains the essential formulas for computing the previously mentioned data.

3.4.3. RINEX Conversion

After calculating the pseudorange, carrier phase, Doppler, and signal-to-noise ratio (SNR) values from the raw GNSS data for all satellites at each epoch, the tool then formats the data into RINEX format. This involves two main steps: first, creating the header, which includes pertinent information such as the capturing app, observables for each constellation in the file, and the initial calculation epoch. Subsequently, the tool generates the observable data section, ordered chronologically, and organized by constellation.

3.4.4. Output

The naming convention for a RINEX file typically incorporates essential information, including the station or site identifier and the commencement time of data collection. Specifically for this tool, the key components of a RINEX 3.05 file name are outlined as follows (Romero I, 2020):

1. Station Identifier: A unique label for the GPS station or site where the data were gathered. It may consist of a combination of letters and numbers, providing a distinctive identification for the specific location. In the output of this tool, there are two distinct Station Identifiers:
   - GNSS00GEA: Used when raw data are recorded using GEA.
   - GNSS00LOG: Applied when raw data are logged using GNSSLogger.

2. Session Start Time: The date and time when the data collection session commenced. Typically expressed in the format YYYYMMDDHHMM, where:
   - YYYY represents the four-digit year,
   - MM denotes the two-digit month,
   - DD stands for the two-digit day,
   - HH signifies the two-digit hour (in UTC), and
   - MM specifies the two-digit minute.

3.5. Software Development and Example Usage

Having introduced the *raw_android_to_rinex* tool and its functionality, we now delve into its practical applications and integration capabilities. The subsequent sections outline the exemplary use of the tool through its command-line interface and shed light on the messages communicated during execution. Following this, we explore an alternative avenue for users seeking more control and integration possibilities by discussing the import library/package aspect of *raw_android_to_rinex*. This modular library empowers developers to seamlessly incorporate GNSS data conversion into their Python projects, offering a customizable approach that aligns with specific project requirements. The discussion in the following sections aims to provide a comprehensive understanding of the software's usability and adaptability, catering to users with diverse needs and preferences. For more details, please refer to Sects. "4. Usage" and "5. Usage for developers" in the attached user manual of the tool, where usage of the tool is explained.

3.5.1. Script with Entry Point

The *raw_android_to_rinex* tool simplifies the conversion of GNSS data to RINEX format through a command-line interface. Users can effortlessly convert data by providing input parameters such as the input GNSS file, output RINEX file, and additional configuration options. The script is designed with an intuitive command-line interface, making it accessible to both beginners and experienced users.

The tool communicates with the user through a set of messages displayed on the console, conveying information about the execution status. There are three main types of messages:

- Info: These messages provide informative details about the status of execution.
- Warning: This type of message is triggered when one of the observables is considered invalid and is therefore excluded from the process. It is crucial to note that this exclusion does not hinder the overall execution. This situation arises when certain values essential for calculating pseudorange, carrier phase, or Doppler are either invalid or recorded inaccurately. Additionally, it can occur when the observable is in an invalid state.
- Error: If an error occurs during execution, this message is displayed, indicating that the tool could not generate results due to the encountered issue.

3.5.2. Import library/package

For users who require more control or wish to integrate GNSS data conversion into existing Python projects, *raw_android_to_rinex* is designed as a modular import library. The library exposes a set of functions that can be seamlessly integrated into custom scripts or applications. By using the *raw_android_to_rinex* library, developers have the flexibility to customize the conversion process according to their specific needs. This approach is particularly beneficial for those who want to incorporate GNSS data conversion as part of a larger workflow or application.

3.6. Data and Materials Availability

The software is available from the GPS Toolbox website at *https://geodesy.noaa.gov/gps-toolbox/*

3.7. Conclusion

Raw_android_to_rinex provides a valuable tool for researchers and developers working with Android GNSS data by facilitating the conversion of raw data to the standard RINEX format. This opens opportunities for integrating Android GNSS data into existing GNSS processing workflows, enabling more comprehensive and interoperable analyses. It also strikes a balance between simplicity and flexibility, offering a convenient script for quick conversions and an importable library for more advanced use cases. Whether you are a casual user needing a straightforward conversion or a developer looking to integrate GNSS data processing into a larger project, *raw_android_to_rinex* provides a reliable and efficient solution. The open-source nature of the toolbox encourages collaboration and further development within the GNSS community.

Capítulo 4

4. Artículo 3: Navigating latency hurdles: an in-depth examination of a cloud-powered GNSS real-time positioning application on mobile devices

Jorge Hernández Olcina, Ana B. Anquela Julián & Ángel E. Martín Furones.
Scientific Reports 14, 14668 (2024).
doi:10.1038/s41598-024-65652-7

4.1. Abstract

A growing dependence on real-time positioning apps for navigation, safety, and location-based services necessitates a deep understanding of latency challenges within cloud-based Global Navigation Satellite System (GNSS) solutions. This study analyses a GNSS real-time positioning app on smartphones that utilizes cloud computing for positioning data delivery. The study investigates and quantifies diverse latency contributors throughout the system architecture, including GNSS signal acquisition, data transmission, cloud processing, and result dissemination. Controlled experiments and real-world scenarios are employed to assess the influence of network conditions, device capabilities, and cloud server load on overall positioning latency. Findings highlight system bottlenecks and their relative contributions to latency. Additionally, practical recommendations are presented for developers and cloud service providers to mitigate these challenges and guarantee an optimal user experience for real-time positioning applications. This study not only elucidates the complex interplay of factors affecting GNSS app latency, but also paves the way for future advancements in cloud-based positioning solutions, ensuring the accuracy and timeliness critical for safety–critical and emerging applications.

4.2. Introduction

The advent of GNSS technology in smartphones has revolutionized the way we perceive and interact with our surroundings. The ability to accurately determine one's position has found applications in numerous fields, from navigation and geolocation services to scientific research and emergency response. The Android operating system has made significant strides in GNSS technology. Prior to the release of Android 7 (Nougat) in 2016, only the position–velocity–time (PVT) computed by the GNSS chipsets was available to users, with positioning accuracy typically between 3 and 5 m. However, with the release of Android 7, raw GNSS measurements, including pseudorange, carrier-phase, Doppler shift, and carrier-to-noise density ratio (C/N0) observations, became accessible, paving the way for more precise positioning applications (Zangenehnejad & Gao, 2021).

The demand for high-precision and reliable positioning systems has pushed the capabilities of receivers to their limits, necessitating innovative strategies to enhance their performance in resource-constrained environments. Efficiently processing GNSS data while grappling with size, energy, and computational constraints is now crucial for propelling positioning and navigation technologies forward. Tackling these challenges will play a pivotal role in shaping the future of GNSS applications and catering to the diverse needs of users across various domains. Therefore, the need to address these challenges efficiently has given rise to the concept of harnessing cloud computing capabilities (Favenza et al. (2014), García-Molina & Parro (2017), Lucas-Sabola et al. (2016), Lucas-Sabola et al. (2018)). Cloud computing offers a solution to the hurdles associated with GNSS data handling and processing within receivers. Additionally, this approach addresses the issue of energy consumption, as many GNSS receivers depend on batteries that drain rapidly under increased computational demands (Lucas-Sabola et al., 2018). Cloud computing provides a promising path for offloading computationally intensive tasks to remote servers, thereby alleviating the load on local devices and optimizing energy utilization. This change in basic assumptions opens new possibilities for more resource-efficient and sustainable positioning and navigation solutions within the GNSS technology landscape.

While scholarly inquiry into cloud computing's application in smartphone-based positioning remains limited, emerging research indicates promising advancements in utilizing this technology for geospatial data analysis and processing. Works by Konstantinos et al. (2013) and Liu et al. (2021), advocate for the integration of cloud computing in geospatial data processing. Konstantinos et al. (2013) proposed a client–server architecture tailored for managing geospatial data, which is adaptable to the context explored in this study. In this framework, the client, represented by a smartphone, collects GNSS data and transmits them to the server for processing, determining the smartphone's position. Similarly, Liu et al. (2021) adopted a comparable methodology, focusing on Real-Time Kinematic (RTK) positioning. Their approach outlines an algorithm for server-side data acquisition and processing, facilitating precise position calculation based on accurate time synchronization. Both studies underscore the potential of cloud computing to optimize geospatial data processing, thus paving the way for more precise and efficient positioning solutions. By harnessing cloud resources, these methodologies address the limitations of client-side capabilities, demonstrating the transformative role of cloud-based methodologies in advancing geospatial applications.

Hence, this study aims to investigate the latency issues encountered in an Android application utilizing GNSS and cloud computing for position calculation. The research focuses on identifying the contributing factors to latency and proposes potential strategies for mitigation. For cloud-based position computation, the RTKLib tool has been selected, inspired by previous studies such as that of Everett et al. (202(), which aimed to optimize RTKLib for measurements conducted with Android smartphones. These studies underscored the need for adjustments to accommodate the suboptimal quality of measurements obtained from mobile platforms. By combining cloud computing with high-performance software featuring intricate algorithms, this approach holds promise for enhancing positioning accuracy while alleviating the computational burden on smartphones.

4.3. Methodology

4.3.1. App and cloud computing setup

A meticulously crafted Android application has been developed to facilitate real-time GNSS positioning through cloud-based computation. This application serves a dual function by seamlessly integrating offline data logging and online real-time processing, effectively functioning as a GNSS receiver. The application enables direct access to the smartphone's GNSS sensor via the Android API (GSA, 2016), allowing for the retrieval of raw GNSS data, including pseudoranges, carrier phases, and Doppler measurements. On the server side, the backend infrastructure is built upon the RTKLib (v2.4.3 b34) calculation engine, module RTKNavi (RTKLib, 2013). The communication between the application and server is established through the WebSocket network protocol, chosen for its real-time advantages and unique characteristics (Olcina et al., 2023).

This vital communication link, described in *Figura 12*, is widely acknowledged as the most effective method for real-time applications due to its distinctive advantages. WebSockets are favoured in real-time communication for several compelling reasons. Firstly, they boast low latency and support bi-directional communication, ensuring swift and responsive data exchange. Additionally, they are efficient and lightweight, optimizing resource usage without compromising performance. Their ability to maintain continuous connections facilitates seamless real-time updates, enhancing user experience. Moreover, WebSockets offer scalability, accommodating growing demands effortlessly. Their event-driven architecture further enhances responsiveness, enabling applications to react promptly to changes. Finally, their robust security measures safeguard data integrity and confidentiality, making them a reliable choice for diverse communication needs (WebSockets, 2024).

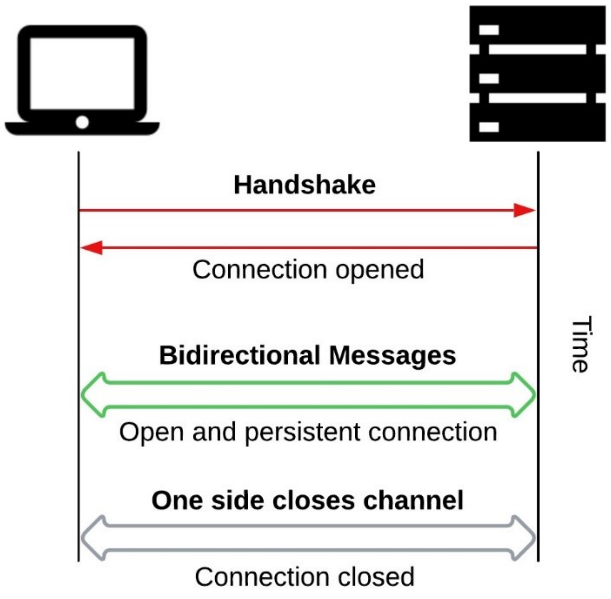


Figura 12: WebSocket connection diagram.

As illustrated in *Figura 13*, the application functions as a GNSS receiver, it captures raw data from the smartphone's sensor, including satellite Pseudoranges, Carrier phases and Doppler measurements, and initiates the WebSocket connection to the server. This connection prompts the server to prepare for execution and launch an RTKLib instance. Concurrently, bidirectional TCP/IP connections are established to facilitate efficient data exchange. Upon connection establishment, the application sends a "start" message, flagging the server's readiness for data collection and processing. Once the raw GNSS data are received, the synchronized server conducts a crucial preprocessing step by converting raw GNSS data into the widely used RTCM3 format (Radio Technical Commission for Maritime Services—Version 3), essential for real-time differential GNSS applications (Radio Technical Commission for Maritime Services (RTCM), 2013).

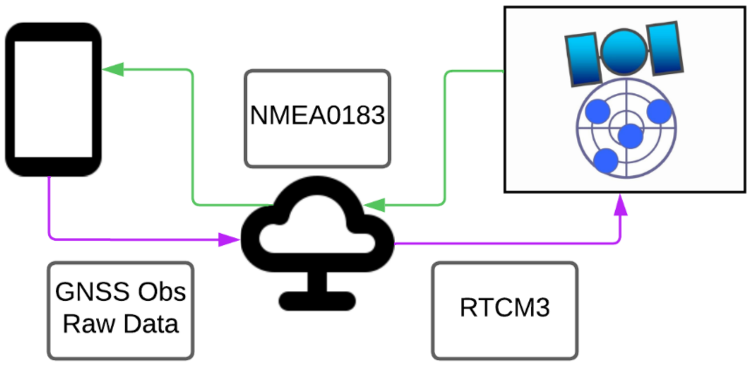


Figura 13: Cloud computing platform workflow.

Converting to RTCM3 format involves a sequence of computations to derive crucial measurements such as Pseudorange, Carrier Phase, Doppler, and temporal values. As outlined by European Global Navigation Satellite System (GSA, 2016), obtaining these measurements directly from the Android API necessitates several steps. First, GPS Time Generation is required, which, in the context of Android 7 or higher, involves utilizing an internal hardware clock and adjusting it to approximate GPS time in nanoseconds, provided the receiver has synchronized with the GNSS reference time. Despite potential delays caused by latency in receiving observations on the server, the position calculation is performed based on the reception time or the time at which the raw GNSS data is received by the smartphone. This ensures that the position calculation is synchronized with the timing of the received data, enabling accurate and timely determination of the user's position despite any transmission delays. Second, Pseudorange Generation involves computing pseudoranges based on temporal disparities between reception and transmission instances, utilizing essential parameters available from the Android system. Third, Carrier Phase Measurements are obtained as *AccumulatedDeltaRangeMeters* (ADRM) represented in meters. Finally, Doppler Measurements are derived from the *PseudorangeRateMetersPerSecond* parameter, with caution required as it represents an "uncorrected" value, lacking adjustments for receiver and satellite clock frequency discrepancies. A positive "uncorrected" value indicates satellite motion away from the receiver.

Once the previously delineated computations were executed, the transmission of the GNSS raw data to RTKLib was facilitated through the utilisation of the MSM5 (Multiple Signal Messages—Type 5) message from the RTCM3 format, which is designed to carry satellite observations from multiple constellations and frequencies. The converted RTCM3 MSM5 messages, along with necessary navigational information, are then transmitted to RTKLib for precise positioning solution computation. After successful computation, the solution is transmitted back to the server through established TCP/IP connections. The server subsequently forwards the computed solution to the smartphone, which, acting as the end-user interface, stores the solution in plain text files adhering to the National Marine Electronics Association (NMEA0183) format. These files can be readily accessed and interpreted by users for accurate positioning information, effectively demonstrating the real-time capabilities of the system.

In summary, this integrated system exemplifies the effectiveness of a cloud-based approach for real-time GNSS positioning on Android devices. The incorporation of WebSockets enhances communication efficiency, making this architecture a promising solution for applications requiring precise and immediate positioning information.

4.3.2. Latency analysis

In the examination of temporal intervals encompassing the transmission of messages to the server (Data Transmission Latency, DTL), to the reception of positioning solutions in the application (Data Retrieval Latency, DRL) and the computational duration (Cloud Processing Latency, CPL), a specialized NMEA-type message has been incorporated to facilitate timestamp creation.

Example:

- $TIMEST,2,1,270,124,102,547.235,S*73

The structured composition of this message is delineated as follows:

0. Message ID $TIMEST.
1. Total count of messages of this type within the current cycle (one epoch).
2. Sequential message number.
3. Date information (ddmmyy).
4. UTC Time data.
5. Source designation: A (App) or S (Server).
6. Checksum data, invariably initiated with an asterisk (*).

This meticulously designed message format allows for the generation of timestamps, thereby enabling a comprehensive analysis of diverse processing phases. Given the inherent lack of synchronization between the server and the smartphone clock, attempting synchronization poses the risk of introducing passive latency. Consequently, during real-time executions, timestamps are systematically generated to distinguish between those associated with the application and the server, acknowledging the intrinsic temporal disparities. Additionally, it is imperative to note that this methodology addresses the challenges posed by asynchronous clock systems, ensuring a nuanced examination of temporal intricacies in the data transmission and retrieval process.

Considering these factors, there will be two app timestamps incorporating time data synchronized with the smartphone clock. The first timestamp occurs at the message output stage, coinciding with the provision of raw GNSS data. The second app timestamp is registered upon the server's return of messages containing position calculation information. This dual timestamp arrangement provides a comprehensive temporal overview, encompassing the duration from message initiation to position computation.

Conversely, the server will generate two timestamps synchronized with the server clock as well. The first timestamp is marked at the instant the server receives the incoming message, while the second timestamp is recorded when the server dispatches the RTKLib output messages. This dual timestamp approach on the server side is instrumental in capturing and assessing the processing time involved in the entire operation.

4.3.3. Data collection

The evaluation of latency behaviour is undertaken through a series of comprehensive tests designed to ascertain its potential impact on positioning accuracy. Two distinct evaluations have been conducted across varied settings to comprehensively assess the latency effect. The first evaluation involves a static test spanning approximately four hours, at a frequency of 1 Hz, while the second entails two kinematic assessment lasting approximately fifteen minutes, at a frequency of 1 Hz. Both tests adhere to a standardized configuration for hardware and software, described in *Tabla 9* and *Tabla 10* respectively, ensuring consistency and comparability across evaluations.

Tabla 9: Hardware specifications.

Device	Android	Psdorange	ADRM	Systems	Network
Google Pixel 7 Pro	14	Yes	Yes	GPS	- Static:
				GAL	4G+ Vodafone
				GLO	- Kinematic:
				BDS	LTE+ Telekom

Tabla 10: RTKLib configuration (RTKLib, 2013).

Options	Settings Value
Positioning Mode	Single
Frequency	L1
Elevation Mask	15º
Ionosphere Correction	Klobuchar
Troposphere Correction	Saastamoinen
Satellite Ephemeris/Clock	Broadcast
Constellations	GPS, GAL, GLO, BDS

On the one hand, the static test aims to simulate prolonged operational conditions, allowing for a nuanced understanding of latency's influence over extended durations. In this scenario, particular attention is directed towards analysing the change points of the ephemeris dataset. The primary aim is to discern whether alterations in the navigation message prompt discernible peaks in positional error, potentially attributable to the utilization of outdated ephemeris data resulting from latency in computational processing.

On the other hand, kinematic tests are designed to capture real-time dynamics and rapid fluctuations in positioning accuracy, providing insights into the immediate impact of latency. Two distinct kinematic tests were conducted, one involving pedestrian movement and the other conducted while driving, to analyse contrasting real-world scenarios: low and high speed, respectively. The objective is to assess the effect of latency in both scenarios and evaluate the technology's feasibility in situations where real-time solutions are critical. These tests entail a comparative analysis between real-time solutions generated by RTKLib and post-processed solutions derived from raw data collected during live executions. For this purpose, the stored raw GNSS data are processed using the RTKPost module, with the data having been previously converted to the RINEX format (Hernández Olcina et al., 2024b). This comparative methodology establishes a benchmark, with the post-processed solution serving as a reference for evaluating the computational real-time accuracy. Besides, this framework facilitates the quantification of positional discrepancies induced by latency.

Finally, considering the implementation of timestamp latency messages elucidated in the preceding section, a specialized tool has been designed to parse the output file generated by the application, containing real-time execution results. This tool facilitates the generation of a graphical representation, wherein latency results including Data Transmission, Data Retrieval, and Cloud Processing Latency are graphically depicted for each calculation epoch. Furthermore, the tool provides a comprehensive array of statistical analyses aimed at elucidating the intricacies of latency behaviour throughout the execution process. The graphical representation affords a visual understanding of latency dynamics across successive calculation epochs, thereby enabling the identification of any temporal patterns or anomalies in latency metrics. By delineating the temporal evolution of latency components, such as data transmission and cloud processing, the graphical output offers valuable insights into the performance of the system under varying computational loads and network conditions. This tool serves as a vital instrument for comprehensively assessing and monitoring latency dynamics throughout the execution process, thereby empowering stakeholders to optimize system performance and mitigate potential bottlenecks in real-time data processing workflows.

4.4. Results and discussion

4.4.1. Static Test

In the conducted static test, the hardware specified in *Tabla 9* was deployed at a fixed outdoor location, for an approximate duration of four hours. The objective was to investigate the impact of latency on error occurrences during transitions between navigation messages. *Tabla 10* outlines the RTKLib configuration employed for position determination.

Given that satellite positions are computed using orbital propagation based on the Keplerian parameters of the navigation messages over time, concerns arose regarding potential discrepancies resulting from latency-induced delays during ephemeris updates. Real-time results were leveraged to scrutinize positional variations over the test duration. *Figura 14* depicts the comparative analysis, revealing that latency fluctuations did not induce significant deviations in any of the three positional components when comparing the solution in real-time with the solution in post-processing. In addition, *Tabla 11* provides a statistical overview of the disparities, or differences, observed between real-time processing and post-processing methods. The data illustrates that, while maximum peaks exceeding 10 cm may occur, the mean and standard deviation indicate minimal divergence between the results obtained through real-time and post-processing. Despite the expectation of no disparities between the two methods in theory, certain factors can contribute to differences.

- Satellite geometry and signal quality. Differences in satellite geometry and signal quality between real-time and post-processed data can lead to discrepancies in positioning accuracy (Wang et al., 2021).

- Atmospheric conditions. Variations in atmospheric conditions, such as ionospheric disturbances or tropospheric delays, can affect GNSS signals differently in real-time versus post-processing scenarios (Tondaś et al., 2020).
- Receiver clock errors. Inconsistencies in receiver clock parameters for code and carrier phase observations can impact the accuracy of GNSS positioning, especially when using a common clock parameter for both types of measurements (Wang et al., 2021).
- Data processing algorithms. The algorithms used for real-time processing and post-processing may differ, leading to variations in how data is handled and corrected, influencing the final positioning results (Wang et al., 2021).
- Multi-constellation solutions Utilizing multi-constellation solutions, like GPS combined with Galileo, can significantly improve accuracy in post-processing compared to real-time solutions due to enhanced signal availability and redundancy (Wang et al., 2021).

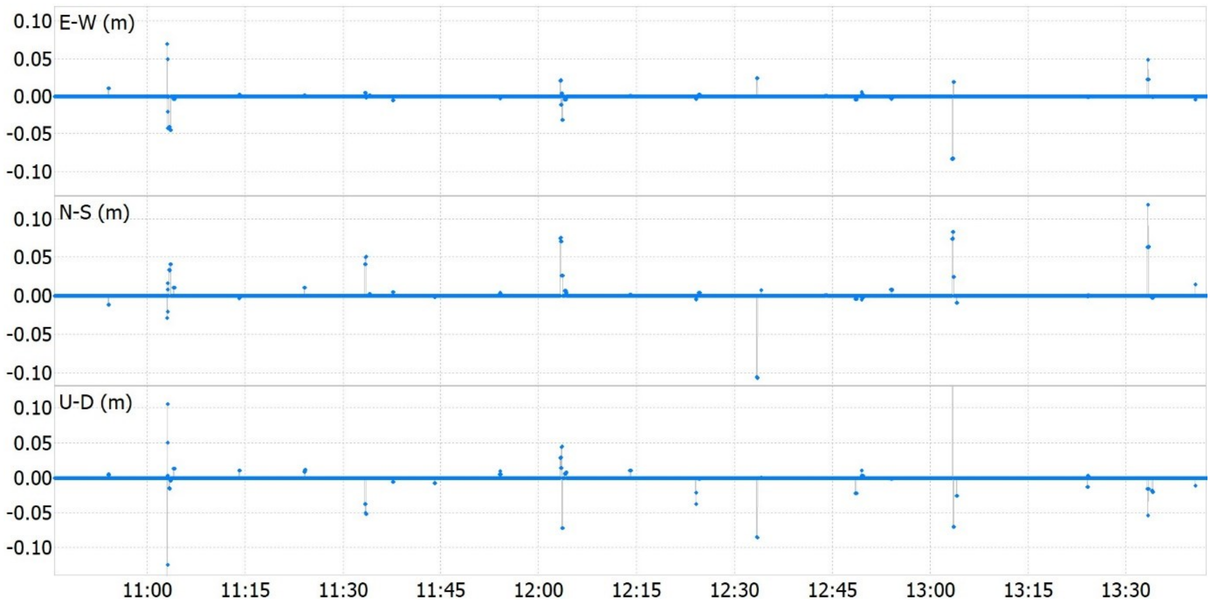


Figura 14: Real-time versus post-processing static test ENU position differences.

Tabla 11: Static test statistics on real-time versus post-processing solution disparities.

	dE (meters)	dN (meters)	dU (meters)
Maximum (abs)	8.3	11.8	17.57
Minimum (abs)	0.0	0.0	0,0
Standard deviation	0.3	0.5	0.7
Mean	0.0	0.0	0.0
Median	0.0	0.0	0.0

Given the minimal disparities observed, attention can be focused on the issue of latency, the communication methodology between the application and server was pivotal in mitigating latency effects, optimizing network capabilities, and yielding latency results well within acceptable thresholds, thereby safeguarding against errors in static positioning. While acknowledging the multifactorial nature of latency, the findings affirm the viability of the communication approach for static position calculations. Consequently, it is concluded that the methodology is promising for static positioning applications.

4.4.2. Kinematic test: *pedestrian*

To assess the impact of latency in kinematic scenarios, a real-world walking test was conducted. Utilizing the hardware and configuration detailed in *Tabla 9*, and the RTKLib setup outlined in *Tabla 10*, a 15-min test was executed under consistent walking conditions. The test route followed an oval trajectory encompassing areas with varied visibility, including wooded regions known to potentially affect positioning solutions.

Comparative analysis was performed between real-time positioning results and post-processed data obtained from the same route. As highlighted in preceding sections, the application not only captures real-time RTKLib outputs but also archives raw sensor data for potential post-processing. *Figura 15* presents the difference between both solutions, revealing negligible positional disparities, often below one millimetre, highlighting the magnitude of the differences between the two methods.

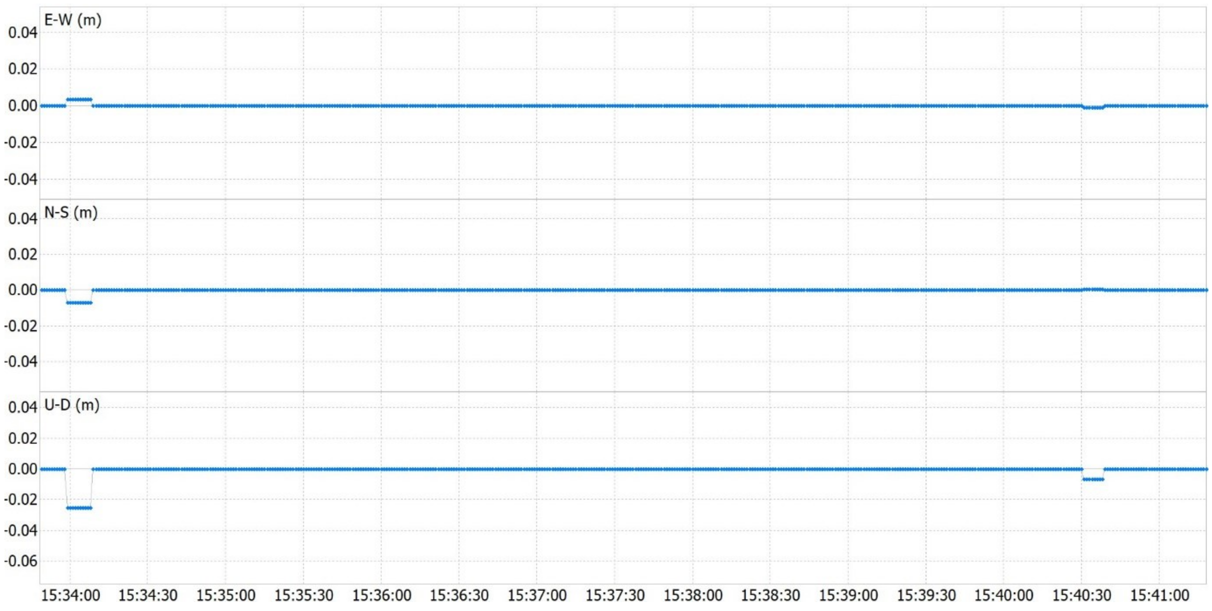


Figura 15: Real-time versus post-processing pedestrian test position differences.

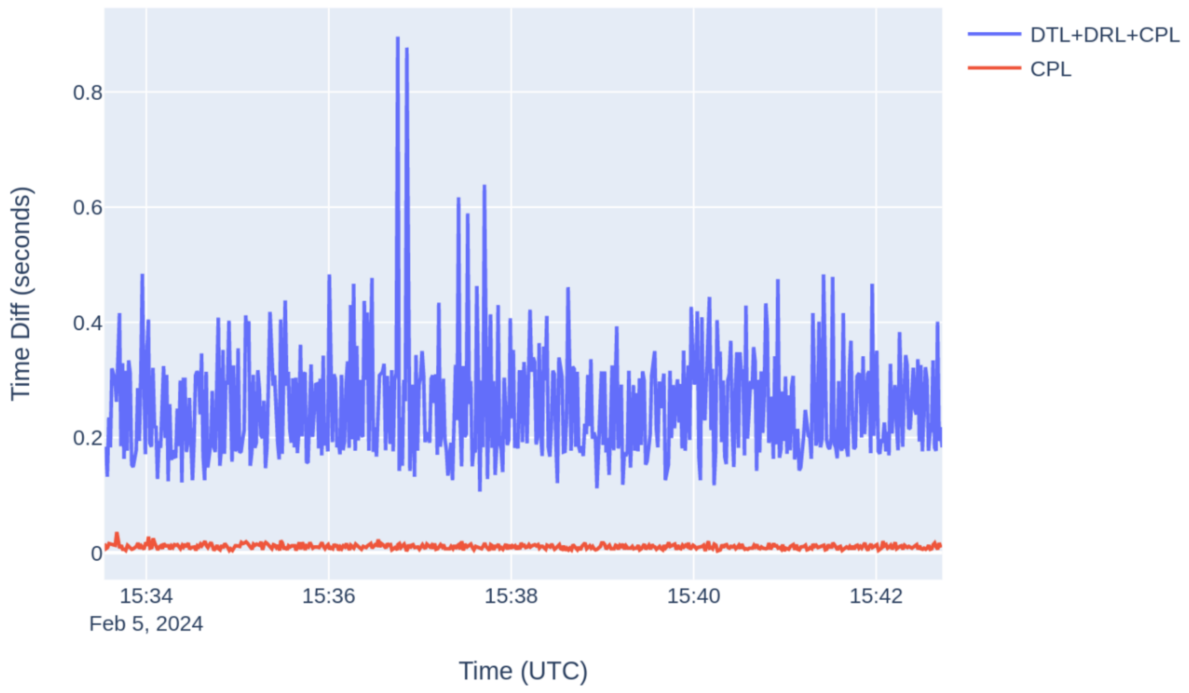


Figura 16: Pedestrian test latency plot.

Tabla 12: Static test statistics on real-time versus post-processing solution disparities.

	DTL+DRL+CPL (seconds)	CPL (seconds)
Maximum	0.8960	0.0360
Minimum	0.1060	0.0030
Standard deviation	0.0947	0.0039
Mean	0.2556	0.0105
Median	0.2240	0.0100

Tabla 13: Pedestrian test statistics on real-time versus post-processing solution disparities.

	dE (cm)	dN (cm)	dU (cm)
Maximum (abs)	0.3	0.7	2.5
Minimum (abs)	0.0	0.0	0.0
Standard deviation	0.0	0.0	0.3
Mean	0.0	0.0	0.0
Median	0.0	0.0	0.0

Figura 16 illustrates the latency metrics derived from the test. The blue line represents the duration between raw data transmission to the server and reception of

positioning results, while the red line denotes server-side processing time, encompassing the duration from data receipt to result dispatch. Additionally, *Tabla 12* provides a concise summary of latency statistics extracted from the graph. The graph and table illustrate the latency distribution within the whole GNSS cloud computing platform. They depict that processing time has the least impact, with most of the latency attributed to transmission time (DTL) and reception time (DRL). This insight underscores the significance of optimizing these phases to enhance overall system performance and user experience.

Analysing the results from *Tabla 13*, minimal differences are once again evident between real-time and post-processing methods, with a maximum peak difference of only 2 cm vertically. These slight disparities may be attributed to numerous factors outlined in the static test. However, they further validate the methodology and affirm that it does not introduce significant errors attributable to latency.

Furthermore, considering the nature of the kinematic test, it is important to acknowledge that due to latency, positions can be displaced depending on the velocity of movement. Assuming an average pedestrian speed of 10 km/h and an average latency of 0.2556 s, the positional variation due to this speed would be approximately 0.7 m. Given the utilization of smartphone-based positioning, it can be inferred that the employed methodology consistently delivers acceptable results regarding latency-induced positional discrepancies.

4.4.3. Kinematic test: *car*

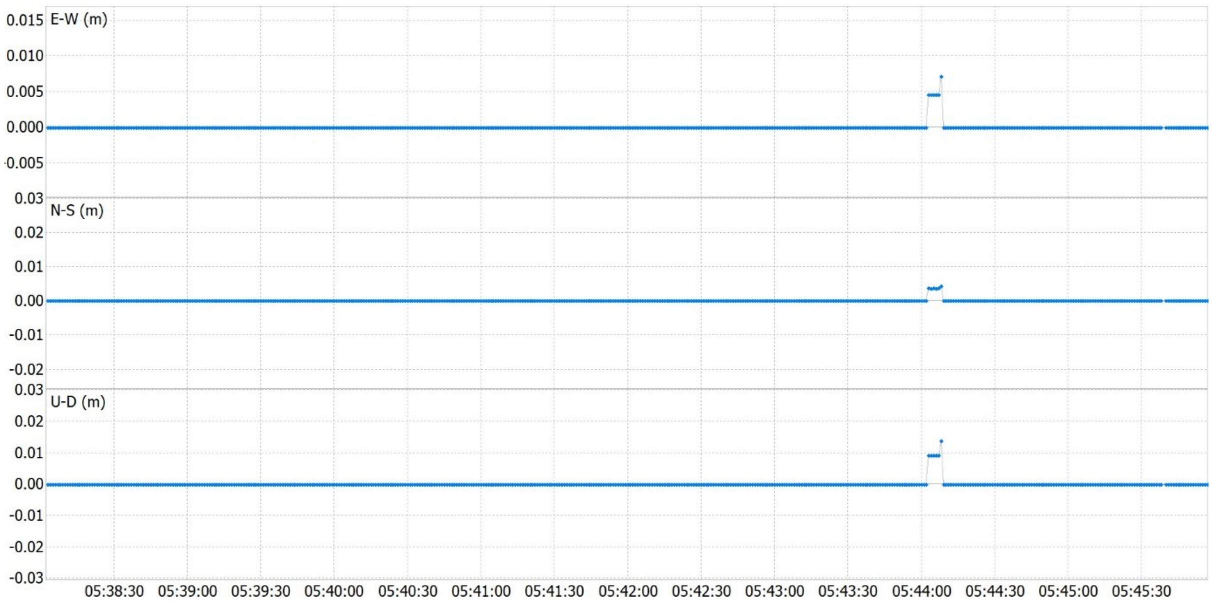


Figura 17: Real-time versus post-processing car test position differences.

To assess the potential negative impact of latency in practical scenarios, a car-based test was conducted, covering approximately 15 km along a highway at an average speed of 120 km/h. Consistent with previous tests, the same hardware and configuration were used, as detailed in *Tabla 9* and *Tabla 10*. The comparative methodology remained consistent, wherein real-time positioning data was juxtaposed with post-processed results.

As depicted in *Figura 17*, positional variations between real-time and post-processing were minimal, echoing findings from prior tests. *Figura 18* illustrates latency results like those observed in previous tests, showing higher DTL, DRL, and CPL. Once more, the data highlights the minimal contribution of CPL to the total latency, with the bulk of the latency primarily stemming from DTL and DRL. This underscores the critical importance of addressing issues related to data transmission and reception to effectively reduce overall latency and improve system efficiency. Furthermore, *Tabla 14* presents latency statistics closely resembling those from preceding experiments. This test has revealed new latency results, which in this instance are higher compared to the previous test, despite it is using the same network. Several factors could contribute to this discrepancy:

- Network congestion. Variations in network traffic can influence the duration of data transmission between the mobile device and the server. During peak periods of congestion, packets may encounter delays as they navigate through the network (Quezada-Gaibor et al., 2022).
- Network latency. The physical distance between the mobile device and the server can introduce latency as data packets travel across the network infrastructure. Additionally, inefficiencies in routing or congestion along the network path can further contribute to fluctuating latency (Quezada-Gaibor et al., 2022).
- Serialization and deserialization Encoding and decoding data sent over WebSockets. The choice of data format (JSON) and serialization library can impact performance (Quezada-Gaibor et al., 2022).
- Server processing time. While the graph in *Figura 18* illustrates constant processing time on the server, it is important to consider potential variations due to factors like resource competition, software optimizations, or background tasks running on the server (Allouch et al., 2021).
- WebSocket performance. Despite offering a persistent, low-latency connection between the client (smartphone) and the server, WebSocket performance can still be influenced by factors such as network conditions and server responsiveness. Variability in WebSocket performance may contribute to latency fluctuations (WebSockets, 2024).
- Device performance. The performance of the smartphone itself can impact app latency. Factors such as CPU usage, available memory, and background

processes on the device can affect the speed of data transmission and reception (Quezada-Gaibor et al., 2022).

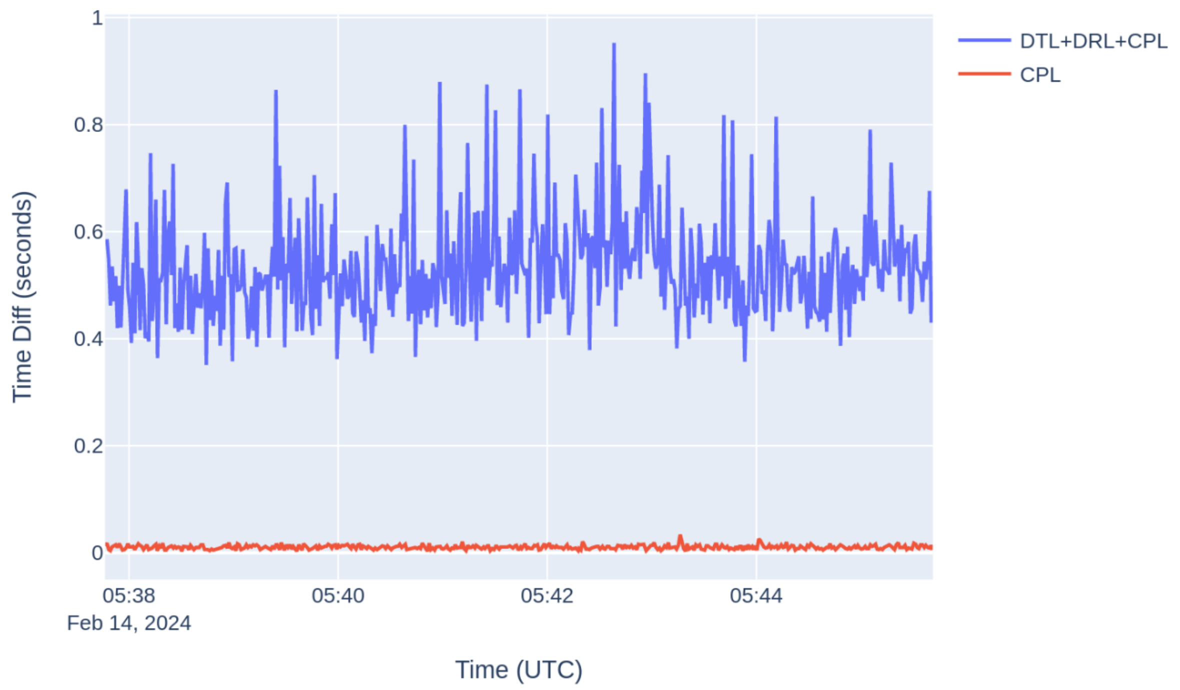


Figura 18: Car test latency plot.

Tabla 14: Static test statistics on real-time versus post-processing solution disparities.

	DTL+DRL+CPL (seconds)	CPL (seconds)
Maximum	0.9530	0.0340
Minimum	0.3510	0.0030
Standard deviation	0.0965	0.0038
Mean	0.5313	0.0105
Median	0.5210	0.0100

Considering an average latency of 0.5313 s at a speed of 120 km/h, the resulting positional variation attributed to latency averages around 17 m. At these speeds, the positional variation becomes significantly greater. Therefore, in scenarios where latency is higher, it can result in a substantial displacement of the vehicle’s position by the time the last calculated position is received. This implies that, depending on the application, such discrepancies could be critical.

Once more, *Tabla 15* presents statistics comparing the three position components between real-time and post-processing methods. As observed in previous cases, although occasional peaks of a few centimetres may occur, the disparity in position between real-time and post-processing remains minor. This reaffirms the validity of the methodology even under high-speed conditions.

Tabla 15: Car test statistics on real-time versus post-processing solution disparities.

	dE (cm)	dN (cm)	dU (cm)
Maximum (abs)	0.7	0.4	1.3
Minimum (abs)	0.0	0.0	0.0
Standard deviation	0.1	0.0	0.1
Mean	0.0	0.0	0.0
Median	0.0	0.0	0.0

As a result, this test highlights that at high speeds, even slight increases in latency values can lead to significant positional variations. This suggests that the methodology may not be suitable for applications where security is paramount, despite yielding results. However, it also underscores the potential for further research aimed at reducing latency within the same methodology, thereby opening avenues for improvement and innovation.

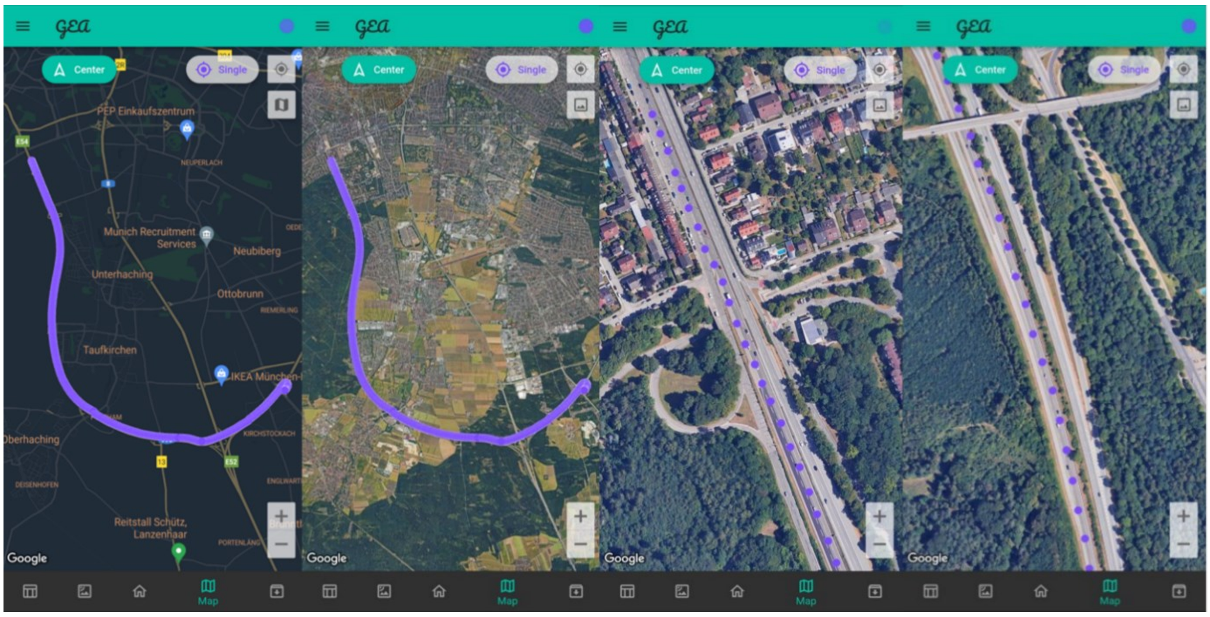


Figura 19: Car test real-time solution snapshots. Maps generated using GEA application version 2.1.0.

Figura 19 offers a comprehensive visual overview, featuring a sequence of screenshots extracted from the application, depicting the test route conducted on a highway by car. These displayed screenshots furnish detailed insights into the performance of the positioning solution. While the primary intent of these findings is not to assess the accuracy of the solution, given its susceptibility to inherent displacement due to speed, they serve to highlight the continuous nature of the solution. The trajectory exhibits smooth transitions without abrupt deviations.

Moreover, the minimal presence of significant outliers attests to the robustness of the employed positioning algorithm. Notably, the solution's accuracy is commendable, as manifested by the consistent alignment of the route with the actual path traversed, notwithstanding the displacement induced by speed and latency. In summary, the visual representation delineated in *Figura 19* underscores the efficacy and dependability of the application in producing precise positioning solutions under dynamic scenarios, such as vehicular travel on roads. However, it is worth noting, as previously mentioned, that movement at elevated speeds can induce positional shifts attributable to latency.

4.4.4. Addressing latency concerns and the viability of cloud computing in GNSS applications

The feasibility and practicality of employing cloud computing in applications have been subject to scrutiny within the scientific community. This scrutiny arises from concerns regarding latency, application scope, and the rationale behind such technological pursuits. In response to these concerns, this section aims to elucidate the rationale behind this investigation and shed light on the viability of cloud powered GNSS applications.

This research is driven by mounting concerns over latency challenges in cloud powered GNSS applications, particularly in contexts where real-time positioning is vital. Latency poses significant hurdles in scenarios necessitating precise and timely positioning. Such concerns have spurred thorough investigation into the impact of latency in practical, real-world settings.

This study explored how latency affects a cloud powered GNSS real-time positioning application, focusing specifically on mobile devices. Through thorough experimentation and analysis, it was found that the impact of latency varies depending on the device's speed. At lower speeds, latency had minimal effect on positioning accuracy. However, as the device's velocity increased, latency-induced position variations became more significant, emphasizing the importance of effective latency management in high-speed applications.

4.4.5. Addressing latency concerns and the viability of cloud computing in GNSS applications

Cloud computing plays a crucial role in addressing challenges related to energy consumption and limited computational capabilities in user devices. By harnessing the computational resources and scalability provided by the cloud, real-time positioning systems can effectively reduce battery consumption issues while improving accuracy and reliability. Additionally, the inherent flexibility of cloud architectures enables the integration of advanced algorithms and techniques, such as Kalman filters, prediction algorithms, and ionospheric monitoring. This integration enhances the functionality and effectiveness of GNSS applications across various domains.

The utility of cloud powered GNSS applications spans a wide spectrum of domains, encompassing navigation, geolocation services, precision agriculture, disaster management, and beyond. By harnessing the power of cloud computing, stakeholders can unlock new possibilities in location-based services, environmental monitoring, and spatial data analysis. Moving forward, further research efforts should focus on optimizing cloud based GNSS solutions, addressing security and privacy concerns, and exploring innovative applications to realize the full potential of this transformative technology.

4.5. Conclusion

This study represents a comprehensive exploration into the nuanced latency challenges inherent in a real-time cloud based GNSS positioning app for smartphones. This app sets in motion an interactive operation by connecting to a specialized server through a WebSocket, which efficiently facilitates the transfer and coordination of data. The server, functioning as a central coordinator, triggers the RTKLib software to perform precise positioning calculations and establishes necessary links for data interchange. Specifically, raw GNSS data undergoes conversion into RTCM3 format, acting as a conduit for further analysis. Following this, RTKLib extracts vital measurements, and the outcomes are relayed to the program for secure retention. The setup underscores the integration of mobile technology, instantaneous communication, cloud-based computing, sophisticated processing, and cloud-driven storage, highlighting its capability to propel GNSS-based positioning applications across various scientific domains.

Through the development and analysis of a bespoke real-time GNSS positioning application leveraging cloud computing for data delivery, a thorough investigation was conducted into the numerous factors contributing to latency across the system.

The research meticulously examined the latency landscape, encompassing critical aspects such as GNSS signal acquisition, data transmission, cloud processing, and the dissemination of results. Utilizing a combination of controlled experiments and real-world scenarios, the study systematically evaluated the impact of network conditions, device capabilities, and cloud server load on overall positioning latency, providing invaluable insights into the intricate interplay of these factors.

By identifying and analysing system bottlenecks, the study offers a nuanced understanding of their relative contributions to latency, thereby providing a roadmap for addressing these challenges. Furthermore, practical recommendations were formulated for developers and cloud service providers to mitigate latency issues and optimize user experience in real-time positioning applications.

The study demonstrates that the precision between the real-time solution and post-processing remains unaffected by latency times. However, the accuracy of the solution in real kinematic scenarios is contingent upon both the velocity at which the smartphone moves and the total processing latency. Based on the experiments conducted, this total latency can range from an average of 0.25 s for users walking to 0.5 s for users traveling at a speed of 120 km/h.

The significance of this research extends beyond its immediate findings, as it lays the groundwork for future advancements in the field. By implementing the recommendations proposed in this study, stakeholders can enhance the accuracy, timeliness, and reliability of real-time positioning systems based on cloud computing, meeting the stringent requirements of safety–critical and emerging applications reliant on instantaneous positioning data.

This study serves as a contribution to the ongoing discourse surrounding real-time GNSS positioning applications, offering actionable insights that have the potential to drive innovation and improve the functionality of such systems in diverse settings and applications.

4.6. Data availability

The datasets generated and/or analysed during the current study are available in the GitHub repository. (*jorgeho1995*)

Capítulo 5

5. Discusión general de los resultados

La presente tesis doctoral ha abordado el desafío de mejorar la precisión y eficiencia del posicionamiento en tiempo real mediante dispositivos móviles a través de la integración de la computación en la nube con sistemas GNSS. A lo largo de los capítulos que componen este compendio, se han presentado tres artículos que desarrollan y validan una metodología innovadora para superar las limitaciones actuales de los dispositivos móviles en el procesamiento de datos GNSS.

5.1. Contribuciones clave y resultados obtenidos

5.1.1. Metodología para la computación en la nube aplicada al posicionamiento en tiempo real

Uno de los logros más significativos de esta tesis es el desarrollo de una metodología que permite utilizar la computación en la nube para el procesamiento en tiempo real de datos GNSS capturados por dispositivos móviles. La implementación de esta metodología ha demostrado ser efectiva para superar las limitaciones inherentes al hardware de los smartphones, tales como la capacidad limitada de procesamiento y el consumo energético elevado.

El segundo artículo presentado, centrado en la conversión de datos GNSS crudos a formato RINEX, ha sido fundamental para establecer una base sólida en la validación de los resultados obtenidos. Esta herramienta no solo permite la comparación de datos procesados en tiempo real con datos procedentes del postproceso, sino que también garantiza que las mediciones puedan ser analizadas con un alto grado de precisión. Los resultados mostraron que, a través de esta metodología, se puede lograr una precisión en tiempo real comparable con la obtenida mediante métodos tradicionales de postproceso, lo cual es un avance significativo en el campo de la geolocalización.

5.1.2. Plataforma de almacenamiento y computación en la nube

El primer artículo en este compendio aborda la implementación de una plataforma de computación en la nube que soporta el procesamiento de datos GNSS en tiempo real. Esta plataforma no solo permite el procesamiento eficiente de grandes volúmenes de datos, sino que también asegura que los resultados sean devueltos a los dispositivos móviles de manera rápida y precisa.

La relevancia de esta plataforma radica en su capacidad para manejar múltiples usuarios de manera simultánea, garantizando que los servicios de posicionamiento en tiempo real sean escalables y robustos. Los experimentos realizados mostraron que la plataforma puede gestionar eficientemente las tareas de procesamiento, minimizando la latencia y maximizando la precisión, incluso en entornos de alta demanda.

5.1.3. Análisis de latencias y viabilidad de la metodología

El tercer artículo se centra en uno de los desafíos más críticos para las aplicaciones GNSS en tiempo real: la latencia. El análisis detallado de las latencias introducidas por la computación en la nube ha permitido identificar las principales fuentes de retraso en el proceso de transmisión, procesamiento y retorno de datos.

Los resultados indican que, aunque la computación en la nube introduce cierta latencia en el sistema, esta puede ser mitigada a través de la optimización de los protocolos de comunicación y la selección adecuada de ubicaciones de servidores en la nube. En el análisis realizado, se observó que la latencia introducida por la computación en la nube puede tener un impacto en la precisión del posicionamiento, especialmente en aplicaciones que implican altas velocidades de movimiento. En estos escenarios, la latencia puede causar un desplazamiento de la posición calculada, afectando la viabilidad del sistema en tiempo real. Sin embargo, a bajas velocidades de desplazamiento, este efecto es mucho menos significativo, y la metodología propuesta sigue siendo viable para aplicaciones como la navegación urbana y el uso en servicios que no requieren reacciones inmediatas a cambios rápidos de posición.

5.2. Evaluación crítica y comparación con el estado del arte

La metodología propuesta en esta tesis ha sido evaluada en comparación con las soluciones existentes en el campo de la geolocalización y el procesamiento GNSS. En términos de precisión, los resultados obtenidos son competitivos con los sistemas de posicionamiento que utilizan hardware especializado, lo que demuestra la viabilidad de los dispositivos móviles como herramientas precisas para aplicaciones GNSS, siempre y cuando se apoyen en la computación en la nube.

En cuanto a la latencia, aunque los sistemas tradicionales que operan completamente en hardware dedicado pueden ofrecer tiempos de respuesta ligeramente mejores, la metodología basada en la nube propuesta aquí ofrece una compensación favorable en términos de flexibilidad, escalabilidad y reducción de costos. Además, la posibilidad de implementar algoritmos más sofisticados y precisos en la nube, que no serían viables en dispositivos móviles, ofrece una ventaja significativa sobre las soluciones existentes.

5.3. Limitaciones y áreas de mejora

A pesar de los resultados positivos, existen ciertas limitaciones en la metodología propuesta que deben ser reconocidas. La dependencia de una conexión estable a Internet es uno de los principales desafíos, ya que cualquier interrupción en la conectividad podría afectar la precisión y la disponibilidad del servicio de posicionamiento en tiempo real.

Asimismo, aunque la latencia ha sido minimizada a niveles aceptables para la mayoría de las aplicaciones, existen escenarios altamente críticos donde cualquier retraso, por pequeño que sea, podría ser problemático. Por lo tanto, futuras investigaciones podrían centrarse en la reducción adicional de latencias a través de mejoras en la infraestructura de red y la optimización de los procesos de transmisión de datos.

Otra área de mejora es la optimización del consumo energético en los dispositivos móviles durante el uso de esta metodología. Aunque la computación en la nube reduce la carga de procesamiento en los dispositivos, el uso continuo de la conexión de datos y la transmisión constante de información GNSS pueden tener un impacto significativo en la duración de la batería. La investigación futura podría explorar formas de hacer que este proceso sea más eficiente energéticamente

5.4. Implicaciones para la práctica y futuras líneas de investigación

Los resultados de esta tesis tienen importantes implicaciones prácticas, especialmente en el desarrollo de aplicaciones móviles avanzadas que requieren un posicionamiento preciso y en tiempo real. Las metodologías y plataformas desarrolladas aquí pueden ser adaptadas y mejoradas para su uso en una amplia gama de contextos, desde la navegación y el transporte hasta la agricultura de precisión y la gestión de recursos en ciudades inteligentes.

Futuras investigaciones podrían explorar la integración de esta metodología con otras tecnologías emergentes, como el 5G, para mejorar aún más la velocidad y la confiabilidad de los servicios GNSS en tiempo real. Además, la extensión de esta metodología a otros dispositivos IoT (Internet of Things) con capacidades limitadas podría abrir nuevas posibilidades en el campo de la geolocalización.

En conclusión, la presente tesis ha demostrado que es posible mejorar significativamente el rendimiento del posicionamiento GNSS en dispositivos móviles mediante la integración con la computación en la nube. Los artículos presentados han abordado de manera integral los desafíos asociados con esta integración, y los resultados obtenidos sientan una base sólida para futuras investigaciones y desarrollos en este campo.

Capítulo 6

6. Conclusiones

La presente tesis doctoral ha explorado y demostrado la viabilidad de una metodología innovadora que integra la computación en la nube con sistemas GNSS para mejorar el posicionamiento en tiempo real mediante dispositivos móviles. A través de un compendio de artículos, se han abordado los principales desafíos relacionados con el procesamiento de datos GNSS en dispositivos con capacidades limitadas, proponiendo soluciones que aprovechan el poder de la computación en la nube para superar estas limitaciones. Una de las contribuciones más significativas de esta tesis es la creación de una metodología que permite el procesamiento eficiente y preciso de datos GNSS en tiempo real mediante el uso de la computación en la nube. Esta metodología ha demostrado ser capaz de superar las limitaciones de los dispositivos móviles, como la capacidad de procesamiento y el consumo de energía, logrando una precisión comparable a la de sistemas de hardware especializado.

Otra contribución clave es la implementación de una plataforma de almacenamiento y procesamiento en la nube que soporta múltiples usuarios y gestiona grandes volúmenes de datos GNSS en tiempo real. Esta plataforma no solo es escalable y robusta, sino que también optimiza la transmisión y procesamiento de datos, minimizando la latencia y garantizando la disponibilidad continua del servicio de posicionamiento.

La investigación también ha proporcionado un análisis exhaustivo de las latencias introducidas por la computación en la nube, identificando las principales fuentes de retraso y proponiendo estrategias efectivas para mitigarlas. Esto ha sido crucial para garantizar que la metodología sea viable para aplicaciones que requieren tiempos de respuesta rápidos, como los sistemas de navegación en tiempo real y los servicios de emergencia.

Los resultados obtenidos en esta tesis tienen un impacto significativo en el campo de la geolocalización, ofreciendo nuevas posibilidades para el uso de dispositivos móviles en aplicaciones que requieren un alto grado de precisión y eficiencia en tiempo real. La integración de GNSS con la computación en la nube no solo mejora la precisión del posicionamiento, sino que también facilita la adopción de estas tecnologías en una amplia gama de aplicaciones, desde la gestión urbana hasta la agricultura de precisión y los sistemas de transporte inteligente.

A pesar de los logros alcanzados, la tesis también reconoce ciertas limitaciones inherentes a la metodología propuesta. La dependencia de una conexión de red estable para la transmisión de datos a la nube es un desafío significativo, especialmente en áreas con cobertura de red limitada o inestable. Además, aunque se han logrado avances importantes en la reducción de latencias, en aplicaciones extremadamente críticas puede ser necesario seguir optimizando estos tiempos de respuesta.

Otro aspecto que considerar es el consumo energético en los dispositivos móviles durante la transmisión y recepción de datos GNSS. Si bien la computación en la nube reduce la carga de procesamiento local, la constante necesidad de conectividad y transmisión de datos puede afectar la autonomía de los dispositivos.

Este trabajo sienta las bases para una serie de futuras líneas de investigación. En particular, se podrían explorar las siguientes áreas:

- **Mejora de la Eficiencia Energética:** Investigar formas de optimizar el consumo energético en dispositivos móviles durante el uso de aplicaciones GNSS en tiempo real, quizás mediante la implementación de protocolos de transmisión más eficientes o el uso de energías renovables en las estaciones base.
- **Integración con Tecnologías Emergentes:** La adopción de tecnologías como el 5G y la inteligencia artificial podría mejorar aún más la velocidad y la precisión del procesamiento GNSS en tiempo real, permitiendo aplicaciones más avanzadas y complejas.

- **Aplicación en Entornos Desafiantes:** Ampliar la investigación para incluir escenarios donde la conectividad es limitada o donde las condiciones ambientales dificultan el uso de GNSS, como en áreas rurales o subterráneas, podría ser un área de gran interés.
- **Extensión a Otros Dispositivos IoT:** Explorar la aplicación de esta metodología a dispositivos IoT con capacidades limitadas podría abrir nuevas posibilidades en campos como la logística, el seguimiento de activos y el monitoreo ambiental.

En conclusión, esta tesis ha demostrado que la combinación de GNSS con la computación en la nube es una estrategia efectiva para mejorar el posicionamiento en tiempo real en dispositivos móviles. Los resultados obtenidos no solo validan la viabilidad de esta metodología, sino que también establecen un marco para futuras innovaciones en el campo de la geolocalización. A medida que la tecnología continúa avanzando, es probable que las soluciones propuestas en esta tesis jueguen un papel importante en la próxima generación de aplicaciones basadas en la ubicación, contribuyendo al desarrollo de infraestructuras inteligentes y a la mejora de los servicios basados en la geolocalización en todo el mundo.

Bibliografía

Allouch, A., Cheikhrouhou, O., Koubâa, A., Toumi, K., Khalgui, M., & Nguyen, G. T. (2021). UTM-chain: Blockchain-based secure unmanned traffic management for internet of drones. *Sensors., 21*. https://doi.org/10.3390/s21093049

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

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

Dutta, D., Mahato, S., Raja, S., Ganguli, S., Verma, S., & Bose, A. (2020). Android Smartphones for GNSS studies in Multi-Constellation Environment. *2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA)*, 1–3. https://doi.org/10.1109/NCETSTEA48365.2020.9119916

Everett, T., Taylor, T., Lee, D. K., & Akos, D. M. (2022). Optimizing the use of RTKLIB for smartphone-based GNSS measurements. *Sensors, 22*. https://doi.org/10.3390/s22103825

Favenza, A., Rossi, C., Pasin, M., & Dominici, F. (2014). A cloud-based approach to GNSS augmentation for navigation services. *Proceedings of the 7th International Conference on Utility and Cloud Computing*.

Firebase. (2023). *Firebase*. https://firebase.google.com

Flutter. (2023). *Flutter*. https://flutter.dev

García-Molina, J. A., & Parro, J. M. (2017). Cloud-based GNSS processing of distributed receivers of opportunity: Techniques, applications and data-collection strategies. *6th International Colloquium on Scientific and Fundamental Aspects of GNSS/Galileo*.

GSA. (2016). *Using GNSS raw measurements on android devices*. Publications Office of the European Union. https://doi.org/10.2878/449581

Hernández Olcina, J., Anquela Julián, A. B., & Martín Furones, Á. E. (2024a). Navigating latency hurdles: an in-depth examination of a cloud-powered GNSS real-time positioning application on mobile devices. *Scientific Reports*, *14*(1), 14668. https://doi.org/10.1038/s41598-024-65652-7

Hernández Olcina, J., Anquela Julián, A. B., & Martín Furones, Á. E. (2024b). Python toolbox for android GNSS raw data to RINEX conversion. *GPS Solut., 28*. https://doi.org/10.1007/s10291-024-01631-9

Konstantinos, E., Konstantinos, N., Stathis, M., & Constantine, P. (2013). Geospatial services in the cloud. *Comput. Geosci., 63*.

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

Lucas-Sabola, V., Seco-Granados, G., Lopez-Salcedo, J. A., Garcia-Molina, J. A., & Crisci, M. (2016). Cloud GNSS receivers: New advanced applications made possible. *2016 International Conference on Localization and GNSS (ICL-GNSS)*, 1–6. https://doi.org/10.1109/ICL-GNSS.2016.7533852

Lucas-Sabola, V., Seco-Granados, G., López-Salcedo, J. A., García-Molina, J. A., & Hein, G. W. (2018). GNSS IoT positioning: From conventional sensors to a cloud-based solution. *Inside GNSS*. https://insidegnss.com/gnss-iot-positioning-from-conventional-sensors-to-a-cloud-based-solution/

Mahato, S., Dutta, D., Roy, M., Santra, A., Dan, S., & Bose, A. (2024). Common Android Smartphones and Apps for Cost-Efficient GNSS Data Collection: An Overview. *IETE Journal of Research*, *70*(2), 1871–1884. https://doi.org/10.1080/03772063.2022.2164369

Mahato, S., Goswami, M., Kundu, S., & Bose, A. (2023). Single-Baseline Long-Distance RTK using a CLS GNSS Module and Open-Source Software: A Case Study from India. *IETE Journal of Research*, 1–12. https://doi.org/10.1080/03772063.2023.2192424

Mindell, D. A. (2008). *Digital Apollo*. The MIT Press. https://doi.org/10.7551/mitpress/7734.001.0001

NodeJS. (2023). *NodeJS*. https://nodejs.org

O'Brien, F. (Ed.). (2010). *The Apollo Guidance Computer*. Praxis. https://doi.org/10.1007/978-1-4419-0877-3

Olcina, J. H., Julián, A. B. A., & Furones, Á. E. M. (2023). Treatment and analysis of the GNSS signal from smartphones and its applicability to urban mobility. *Environ. Sci. Proc., 28*. https://doi.org/10.3390/environsciproc2023028001

Quezada-Gaibor, D., Torres-Sospedra, J., Nurmi, J., Koucheryavy, Y., & Huerta, J. (2022). Cloud platforms for context-adaptive positioning and localisation in GNSS-denied scenarios—A systematic review. *Sensors., 22*. https://doi.org/10.3390/s22010110

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*.

Raw GNSS Measurements. (2016). *Android Developers*. https://developer.android.com/develop/sensors-and-location/sensors/gnss

Realini, E., Caldera, S., Pertusini, L., & Sampietro, D. (2017). Precise GNSS positioning using smart devices. *Sensors, 17*. https://doi.org/10.3390/s17102434

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*, *8*. https://doi.org/10.3390/electronics8010091

Rokubun. (2020). *Android GNSS Logger to RINEX Converter*. https://github.com/rokubun/android_rinex

Romero I. (2020). *RINEX The receiver independent exchange format version 3.05*. IGS/RTCM RINEX WG Chair ESA/ESOC/Navigation Support Office. https://files.igs.org/pub/data/format/rinex305.pdf

RTKLib. (2013). *RTKLib ver. 2.4.2 Manual*. Https://Www.Rtklib.Com/Prog/Manual_2.4.2.Pdf. https://www.rtklib.com/prog/manual_2.4.2.pdf.

Tondaś, D., Kapłon, J., & Rohm, W. (2020). Ultra-fast near real-time estimation of troposphere parameters and coordinates from GPS data. *Measurement*, *162*. https://doi.org/10.1016/j.measurement.2020.107849

Wang, L., Li, Z., & Wang, N. (2021). Real-time GNSS precise point positioning for low-cost smart devices. *GPS Solut.*, *25*. https://doi.org/10.1007/s10291-021-01106-1

WebSockets. (2024). *WebSockets Standard*. https://websockets.spec.whatwg.org

Zangenehnejad, F., & Gao, Y. (2021). GNSS smartphones positioning: Advances, challenges, opportunities, and future perspectives. *Satell. Navig.*, *2*. https://doi.org/10.1186/s43020-021-00054-y

Zangenehnejad, F., Jiang, Y., & Gao, Y. (2023). GNSS observation generation from smartphone android location API: performance of existing apps. *Issues Improv Sens*, *23*. https://doi.org/10.3390/s23020777