ISBN 978-987-4417-90-9

Arquitectura Cliente-Servidor de Alto Rendimiento para servicio RTK

José H. Moyano[1,2], Karina M. Cenci[1,2], and Jorge R. Ardenghi[1,2]

[1] Laboratorio de Investigación en Sistemas Distribuidos
[2] Laboratorio de I+D en Ing. de Software y Sistemas de Información (UNS-CIC Provincia de Buenos Aires)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, Argentina
{jose.moyano,kmc,jra}@cs.uns.edu.ar

Resumen La exactitud de los *Global Navigation Satellite Systems* (GNSS) varía entre 2 y 10 metros. Es posible mejorar la exactitud a nivel de centímetros en tiempo real utilizando *Real Time Kinematic* (RTK). Con esta técnica, un receptor con posición conocida (*base station*), calcula los errores y transmite información de corrección al *rover*, para que se localice con precisión de centímetros. RTKLIB es una biblioteca de software *open source* que permite implementar aplicaciones BS o *rover*. Este trabajo propone la implementación de una arquitectura cliente-servidor de bajo costo, que provea información de corrección en tiempo real a dispositivos *rovers*, utilizando RTKLIB y hardware comercial estándar.

1. Introducción

Global Navigation Satellite Systems (GNSS) es el conjunto de sistemas de georreferenciamiento satelital, donde *space vehicles* (SV) en órbita transmiten información a través de radiofrecuencia a receptores en tierra. Con esta información, el receptor calcula la distancia geométrica a cada SV seguido, y utilizando trilateración, puede localizarse en la superficie terrestre [9].

El error en el posicionamiento es provocado por incertidumbre en las órbitas y *clocks* de los SV; retrasos en la señal provocados por la atmósfera [7]; y ruido y *multipath* en el receptor [5]. Por este motivo, se logra una exactitud horizontal de entre 6 y 10 metros [17] para receptores de simple constelación y simple frecuencia, y de entre 2 y 6 metros para receptores con soporte para múltiples constelaciones o frecuencias. Estas precisiones resultan insuficientes para tareas topográficas y geodésicas, por lo que existen distintos aumentos [9] que mejoran el desempeño.

Real Time Kinematic (RTK) es una técnica que utiliza diferencias simples y dobles sobre las medidas de fase de las señales satelitales capturadas por dos receptores, resuelve la ambigüedad de ciclos de la portadora utilizando algoritmos como LAMBA, y logra posiciones con centímetros [15, 12, 10] de exactitud en tiempo real.

RTK requiere la instalación de un segundo dispositivo receptor GNSS, de posición conocida, que a partir de la información satelital obtiene parámetros de error en el posicionamiento, y genera información de corrección [6]. A este dispositivo se lo llama *estación base* (BS). Esta información de corrección es transmitida a un *rover*, que es el nombre con el que se designa al objeto de posición desconocida. El *rover* también cuenta con un receptor de GNSS, pero necesita la información de corrección para ajustar su localización y lograr la exactitud buscada.

La información provista por una estación base tiene un rango de utilidad acotado por las condiciones atmosféricas. En la práctica, bajo condiciones ionosféricas promedio, estos errores pueden despreciarse en distancias entre receptores de hasta 10km [14]. Cuanto más lejos se encuentre el *rover* de la estación base, menor precisión tendrán las correcciones. Es por ello que es deseable que la estación base no se encuentre muy alejada del *rover*, y también lo que motiva instalar varias BS si se desea proveer el servicio en un área amplia de terreno.

Las estaciones base fijas de operación ininterrumpida, que se instalan para proveer servicio de corrección RTK de forma permanente, se denominan *Continuos Operation Reference Stations* (CORS). *Network RTK* (NRTK) es la técnica de utilizar una red de CORS, para proveer a *rovers* dentro del área de cobertura de las CORS con información de corrección RTK. Los *rovers* utilizan una conexión a internet, generalmente GSM, para solicitar a un servidor la información de corrección de la CORS más cercana [2].

El precio de adquirir BS particulares, los montos elevados de servicios de CORS, o el costo y disponibilidad acotada de las comunicaciones GSM de alta velocidad para conectar con NRTK, resultan un impedimento para el uso masivo de las tecnologías de posicionamiento de alta precisión, y motivan buscar alternativas de bajo costo.

RTKLIB es una biblioteca de software escrita en lenguaje C con licencia BSD2-clause [8], que ofrece la generación de información de corrección, o el uso de información de corrección para obtener soluciones de posición con centímetros de precisión. Esto permite implementar una BS en el primer caso, o un *rover* en el segundo. Takasu y Yasuda [20] demostraron un desempeño razonable utilizando un módulo u-blox y RTKLIB.

El objetivo de este trabajo es analizar las debilidades en el diseño cliente-servidor de RTKLIB, y proponer mejoras o adaptaciones que pueden aplicarse siguiendo los criterios de aplicaciones cliente-servidor de alto rendimiento.

En **Estructura del software RTKLIB** se revisa el diseño de la biblioteca y los puntos importantes para este trabajo. **Análisis de RTK y RTKLIB como servicio de alto desempeño** evalúa RTK desde la perspectiva cliente-servidor, y si RTKLIB respeta criterios de alto desempeño para una implementación. **Propuesta de arquitectura cliente servidor de alto desempeño para posicionamiento RTK** presenta una propuesta de implementación de servicio. En **Discusión** se presentan las observaciones y resultados de una primera implementación.

2. Estructura del software RTKLIB

RTKLIB ofrece funciones para calcular posicionamiento de precisión con distintas técnicas (PPP, DGPS, RTK), en simple y doble frecuencia, en tiempo real y para pos-procesamiento, para las constelaciones GPS, GLONASS, Galileo, QZSS, BeiDou, SBAS [19]; a través de código fuente C, y adaptado para sistemas POSIX y WIN32.

Entre sus utilidades, se incluyen programas para compilar en Borland C++ para sistemas Windows, que permiten hacer pos-procesamiento con datos de *rover* y BS, y con servicios de NTRIP a través de internet. Otras herramientas permiten hacer *plots* de un recorrido con un mapa como fondo, y convertir distintos formatos de intercambio de mensajes.

RTK: Del conjunto de prestaciones que incluye RTKLIB, es de relevancia para este trabajo la función de RTK, que se encuentra implementada en el archivo fuente `rtksvr.c`.

`rtksvr.c`, entre sus funciones, cuenta con dos rutinas que realizan los cálculos de corrección: `rtksvrstart` y `rtksvrthread`, donde la primera inicia mediante llamadas al sistema (*syscalls*) un hilo de ejecución que corre la segunda.

`rtksvrthread` itera sobre las observaciones satelitales, y para cada observación genera la información de corrección, en secuencia. Cada una de estas observaciones tiene que ser válida a partir de la información de *fix* de GNSS.

`rtksvr.c` requiere otros archivos fuente para implementar las comunicaciones con los dispositivos, operaciones matemáticas, *logging* y exclusión mutua. Para generar los datos corregidos, requiere de `rtkpos.c`, el cual invoca a las funciones que implementan el filtro de Kalman extendido y el algoritmo MLAMBDA.

3. Análisis de RTK y RTKLIB como servicio de alto desempeño

Como arquitectura cliente-servidor de alto rendimiento, RTKLIB puede analizarse desde dos puntos de vista: Las virtudes y falencias que presenta como servicio a clientes, y las propiedades que tiene como proceso que se ejecuta en un servidor.

Análisis como servicio: Como servicio provisto por un servidor de alto desempeño [11], RTKLIB carece de una interfaz adecuada. La biblioteca sólo provee un hilo de ejecución que, utilizando las observaciones satelitales, genera información de corrección o soluciones de posición y la deja disponible en estructuras internas.

Esta implementación, aunque cuenta con la capacidad de ofrecer una solución *open source* a la técnica de RTK, no avanza sobre la creación de un servicio de posicionamiento de alta precisión que pueda abastecer a más de un cliente, sea local o remoto, sin que estos clientes incorporen como parte de su código fuente a la biblioteca, enlacen con ella, y cuenten con acceso a hardware adecuado para realizar y transmitir las correcciones.

Con el advenimiento de los dispositivos móviles y las tecnologías IoT, es de esperar se implemente un sistema específico de BS, formado por un sistema embebido, que responda a solicitudes de información de corrección de más de un *rover*, siendo los *rovers* dispositivos móviles diversos, como pueden ser teléfonos móviles, tabletas, dispositivos de rastreo, *smartwatches*, sensores, sistemas IoT, etc.

Resulta por ello importante proponer una estructura superior de gestión de solicitudes y comunicaciones, que pueda aprovechar los cálculos y el hardware especializado asociado a una BS, para suministrar de forma eficiente, respetando las cotas de tiempo real, información de corrección RTK.

Comunicaciones: La implementación clásica de estación base simple para posicionamiento de precisión, consiste en la BS con su antena GNSS, y una conexión a un *transceiver* de radio. Esta implementación está limitada a *rovers* que cuenten con receptores de radio compatibles con el *transceiver*.

Actualmente, las comunicaciones de radio son digitales, soportando internamente protocolos de red avanzados como TCP/IP. A nivel de capa de enlace, se pueden citar como ejemplos los estándares inalámbricos IEEE 802.11 y 802.16 en todas sus variantes, y el estándar GSM.

En este escenario, una BS moderna que pretenda proveer servicio de RTK a un conjunto de dispositivos de diversas prestaciones, contará con un enlace de red TCP/IP, que posee la característica de ser independiente del medio, y puede recibir solicitudes a través de redes cableadas o inalámbricas.

Es de esperar entonces que un servidor RTK implementado con RTKLIB, tenga una interfaz TCP/IP, y la forma en la que esa interfaz se comunica con sus clientes dependa de la implementación, esto es, el hardware de comunicaciones con el que cuenta el dispositivo que funciona como servidor.

Análisis como proceso: El hilo `rtksvrthread`, procesa cada una de las observaciones del *rover*, generando la posición RTK con la función *rtkpos*. Este procesamiento se realiza en secuencia, procesando una observación luego de la otra de forma sucesiva (figura 1).

```
....
for (i=0;i<fobs[0];i++) { /* for each rover observation data */
    obs.n=0;
    for (j=0;j<svr->obs[0][i].n&&obs.n<MAXOBS*2;j++) {
        obs.data[obs.n++]=svr->obs[0][i].data[j];
    }
    for (j=0;j<svr->obs[1][0].n&&obs.n<MAXOBS*2;j++) {
        obs.data[obs.n++]=svr->obs[1][0].data[j];
    }
    rtksvrlock(svr);
    rtkpos(&svr->rtk,obs.data,obs.n,&svr->nav);
    rtksvrunlock(svr);
....
```

Figura 1. Procesamiento de posiciones satelitales en Segmento `rtksvrthread`

Dado que las observaciones pueden ajustarse independiente una de la otra, el diseño de este algoritmo, no considera aprovechar las posibilidades que ofrece un sistema multiprogramado de tiempo compartido, como es un sistema Windows o Linux para los cuales está preparado, o las posibilidades que ofrecen los procesadores de múltiple núcleo, que se encuentran disponibles incluso en dispositivos simples.

Esta implementación se construye pensando en un hilo de ejecución, que corre como parte de un programa principal, el cual incluye a RTKLIB como parte de su código fuente, y está pensado también para procesamientos *offline* (en el caso de programas de escritorio), o con hardware limitado (para implementaciones en sistemas embebidos).

Análisis de RTK:

Cliente: El cliente de una BS es el responsable de solicitar la información de corrección, procesarla y generar la solución con precisión de centímetros. En un escenario donde un área es cubierta con estaciones base para ofrecer servicio de RTK, también debe escoger la BS adecuada. Involucra determinar BS disponibles en el radio de alcance del *rover*, y escoger la que combine cercanía, calidad de información y menor carga.

Desde la perspectiva de una arquitectura cliente-servidor, podemos decir que el *rover* es un cliente pesado (*fat*) [21], que contiene la lógica de negocio del servicio RTK.

Servidor: Analizando implementaciones comerciales actuales de RTK, el servidor (BS), nunca es *fat*. Las responsabilidades de la estación base están limitadas a generar información de corrección a partir de observaciones y posición conocida. Luego, esta información es transmitida.

Otras características del servidor: En el caso de transmisiones públicas desde el servidor, es conveniente mantener un enlace sin estado. No necesita el servidor conocer al *rover*, sólo responder sus solicitudes.

Cuando la transmisión desea restringirse, como es el caso de información de corrección que se destina a una suscripción paga, o a usos particulares de una organización, es necesario cifrar la comunicación y exigir credenciales de ingreso. En este caso, puede mantenerse la comunicación sin estado con relativa simplicidad, con un *rover* enviando en su solicitud sus credenciales.

4. Propuesta de arquitectura cliente servidor de alto desempeño para posicionamiento RTK

Actualmente, pueden encontrarse en el ambiente científico, comercial y en comunidades tecnológicas, intentos RTK a bajo costo [20, 16]. Estos intentos se concentran en la implementación de una BS que soporte uno o dos clientes, pero no se consideran las necesidades de un sistema que pueda proveer posicionamiento de precisión a un gran número de dispositivos, como son los dispositivos IoT, ni toman en cuenta los requisitos de rendimiento y comunicaciones de los sistemas cliente-servidor que implementan estas funciones.

El objetivo de este trabajo es abordar el próximo paso en la implementación de una solución de RTK, teniendo en consideración una arquitectura cliente-servidor de alto rendimiento que pueda entregar a *rovers* información de corrección confiable, realizando los ajustes que requiera RTKLIB para mejorar su desempeño y soportar múltiples enlaces de comunicación.

4.1. Topología

El uso de RTK con BS simple, corresponde a la topología de cliente único. Es la metodología tradicional para agrimensura y geodesia, con un usuario propietario de la estación base y el dispositivo receptor. El usuario es encargado de la instalación y configuración de ambos elementos.

Esta topología desperdicia la capacidad de una BS de proveer a varios *rovers*; y no se considera la posibilidad de ofrecer servicio RTK a terceros mediante estaciones de referencia permanentes.

En contrapartida, una arquitectura con múltiples clientes [21], se cuenta con una BS como servidor, y cualquier número de clientes pueden conectar con ella para obtener correcciones.

Finalmente, la estructura más adecuada para implementar la arquitectura cliente-servidor RTK, es contar con múltiples clientes y múltiples servidores pensando el servicio como cobertura en un área de estaciones base.

Lo que permite es la escalabilidad del servicio. La calidad de los datos de corrección está limitada por la distancia entre el *rover* y la estación base. Es por este motivo que para ampliar el área de cobertura del servicio, es necesario instalar servidores que puedan ofrecer la información de corrección a clientes cercanos.

4.2. Tiers y arquitectura

Desde el punto de vista de una aplicación cliente-servidor, definimos para RTK:

- **Lógica de aplicación:** Determinar la posición en tiempo real de un *rover* a partir de la información de corrección.
- **Lógica de negocio:** Existiendo más de una BS, consiste en encontrar y escoger la más cercana geográficamente, con mejor calidad de correcciones y disponible; y establecer la conexión.
- **Lógica de datos:** El servicio de BS generando la información de corrección válida para su zona de influencia.

Por lo expuesto anteriormente, una arquitectura de 2T, delega en el cliente seleccionar el servidor más cercano. Esto es, la lógica de negocio debe estar necesariamente en el cliente. A esta responsabilidad se añade también mantener una lista de servidores RTK con sus localizaciones, actualizar esta lista, y determinar cuál es el servidor más adecuado a utilizar.

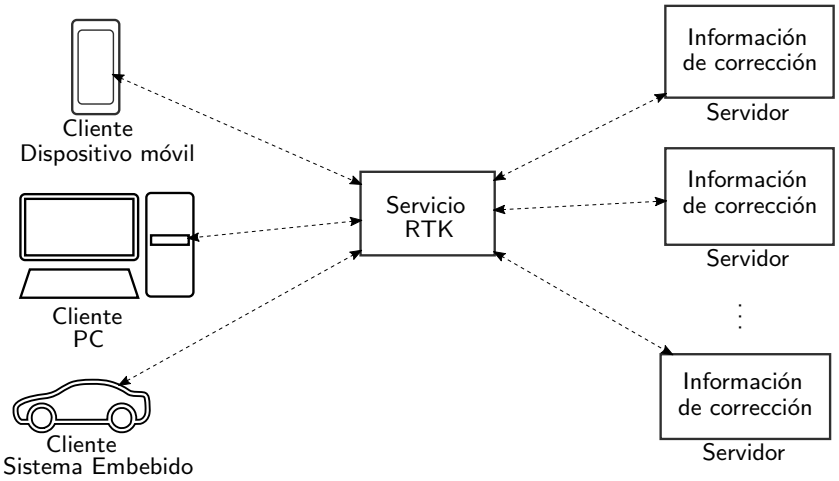

Figura 2. Arquitectura 3T

Estas tareas son demandantes en procesamiento y comunicaciones. Si consideramos que el *rover* es un dispositivo móvil, funcionando a baterías, posiblemente conectado a través de una red de datos móviles paga, la arquitectura 2T no es la más adecuada para proveer un servicio RTK.

Si se implementa una arquitectura 3T (figura 2), el cliente se comunica directamente con un servidor que maneja la lógica de negocio. Éste conoce las BS, su localización y disponibilidad, y sería posible incorporar balance de carga entre ellas. Además, aumenta la flexibilidad pudiendo agregar nuevas estaciones de forma transparente al cliente, y facilita la implementación de conexiones con estado para transacciones más eficientes.

4.3. Implementación

A nivel de lógica de datos, se implementa un servicio BS en Linux, que compila con RTKLIB, abriendo un puerto de comunicaciones TCP/IP. Para la lógica de negocio, se crea un servicio para Linux que se conecta al servicio BS. La función de la lógica cliente es obtener correcciones de la lógica de negocio y enviar a través de una interfaz USB/RS232 a su módulo GNSS. Este módulo tiene soporte para correcciones.

Plataformas: Raspberry Pi 3 Model B+, con procesador de arquitectura ARM Cortex A53, y sistema operativo Linux, distribución Raspbian Buster.

Los receptores son placas de evaluación Smart GPS revisión 1.11. El *chip* de GNSS fue desoldado y reemplazado por u-blox LEA-6T-0 de igual *pinout* con soporte para mensajes *raw* con formato propietario ublox y RTCM. El funcionamiento se validó utilizando el software u-center provisto por el fabricante del módulo, con información de corrección del servicio NTRIP provisto por RAMSAC [13].

Se utilizaron dos modelos de antenas, ambas activas, Garmin GA25MCX y Taoglas AGGP .25F.

Código fuente: Se escribe en lenguaje C versión C89 [1], compilado para la combinación de SO y procesador.

Condiciones de operación: Las primeras pruebas fueron realizadas con el modo *moving baseline* en condiciones acotadas. Está planificada una segunda etapa de pruebas utilizando puntos geodésicos conocidos para las antenas.

5. Discusión

Las pruebas iniciales se realizaron con una PC como *rover* y Raspberry Pi como BS, funcionando con *moving baseline*. `htop` para verificar desempeño de los procesos. La exactitud de las mediciones se determinó midiendo la distancia geométrica entre antenas.

Exactitud: La exactitud entre 2 y 6 metros con baja precisión, no resultó la esperada u obtenida en otras pruebas realizadas sobre RTKLIB [20]. Sin embargo, esto puede estar vinculado al hardware utilizado. Se necesita un análisis más exhaustivo, reducir el *multipath* utilizando planos a tierra para las antenas, utilizar versiones recientes de los módulos GNSS (M8T, M8N), hacer verificaciones con base estacionaria, comparar con medidas de pos-procesamiento.

Procesamiento y comunicaciones: Bajos requisitos de procesamiento y comunicaciones. La carga generada por los hilos de ejecución no resultó significativa a la operación del sistema, así como las comunicaciones. En pruebas posteriores, es necesario simular un volumen alto de solicitudes, a través de un enlace inalámbrico, para verificar los efectos de la interferencia.

Problemas observados: RTKLIB no implementa una arquitectura de software definida. Sufre de *code smells*: Archivos de código fuente extensos, funciones con decenas de parámetros, líneas con múltiples sentencias, código condicional, compilación con *warnings*. Muchas de sus variables tienen nombres de una o dos letras. El autor de este trabajo supone que se pretende relacionar los algoritmos implementados con sus ecuaciones, como son el filtro de Kalman extendido que suaviza la posición, y la implementación de MLAMBA para resolver la ambigüedad. Estas variables de una letra designan matrices de covarianza, actualización de estado, predicción, observaciones, entre otros. El problema de este diseño yace en que el código fuente no cuenta con recursos tipográficos que hagan legibles a estas declaraciones. Deberían reemplazarse con nombres que declaren intencionalidad.

Es recomendable realizar un *refactoring* del código, estableciendo primero *test harnesses* [3], y reemplazar las declaraciones de tipo por `stdint.h`. Las verificaciones realizadas en [20] utilizaron con un procesador de arquitectura de 32 bits, mientras que las implementaciones evaluadas en este trabajo utilizan un procesador con arquitectura de 64 bits, generando incertidumbres en el funcionamiento al no utilizar tipos de datos estandarizados.

Estas modificaciones son difíciles de realizar. RTKLIB cuenta con una batería de tests con un *coverage* limitado, que resultado difícil de evaluar debido a la

falta de un criterio estándar de validación. Estos tests están implementados como programas C independientes que ejecutan fragmentos de código.

6. Conclusiones

Es indiscutible la ubicuidad de sistemas electrónicos y de software en actividades industriales, comerciales y particulares. Desde dispositivos móviles (*smarphones*, tabletas, *smartwatches*, etc.); hasta sistemas de *Internet of Things* (IoT) que actúan y miden condiciones ambientales con algún propósito, como detectar situaciones de riesgo (contaminación, incendios), recopilar información (estaciones meteorológicas), o automatizar vehículos y máquinas. Una de las necesidades frecuentes en estos sistemas es determinar la localización. La tecnología de RTK ofrece posicionamiento y localización con precisión de centímetros en tiempo real, valiéndose sólo de un receptor y antena de GNSS comercial, si se cuenta con un servicio que provea la información de corrección.

Como se mostró en este trabajo, es posible incorporar RTK a sistemas con GNSS, sin un aumento significativo de costo, con versiones alternativas compatibles de los módulos GNSS ya utilizados. Sin embargo, esta tecnología requiere un proveedor de información de corrección. Aunque existen servicios gubernamentales de NRTK, suelen estar restringidos, y tener como objetivo su uso en agrimensura [13]. Además sufren limitaciones de cobertura, por la distancia a la CORS más cercana, o por la ausencia de conexión a internet de alta velocidad.

RTKLIB permite implementar con hardware genérico, dispositivos de bajo costo que generan la información de corrección. En los últimos años, estos elementos han provocado un creciente interés en la comunidad tecnológica y científica con necesidades de RTK de bajo costo [4, 22, 18]. Este trabajo inicia los primeros pasos para la implementación de un servicio RTK escalable, de alta disponibilidad, construido en software libre, que haga realidad el objetivo de posicionamiento satelital de alta precisión para aplicaciones de IoT.

Referencias

[1] Computer y Business Equipment Manufacturers Association. *ISO/IEC 9899:1990*. Inf. téc. International Organization for Standarization, 1990.

[2] Paolo Dabove y col. «Network Real Time Kinematic (NRTK) Positioning – Description, Architectures and Performances». En: *Satellite Positioning - Methods, Models and Applications*. Mar. de 2015, págs. 23-46.

[3] M.C. Feathers. *Working Effectively with Legacy Code*. Martin, Robert C. Prentice Hall PTR, 2004.

[4] María S. Garrido-Carretero y col. «Low-cost GNSS receiver in RTK positioning under the standard ISO-17123-8: A feasible option in geomatics». En: *Measurement* 137 (2019), págs. 168-178.

[5] André Hauschild. «Basic Observation Equations». En: *Springer Handbook of Global Navigation Satellite Systems*. Ed. por Peter J.G. Teunissen y Oliver Montenbruck. 2017. Cap. 19.

[6] William Henning. *User Guidelines for Single BaseReal Time GNSS Positioning*. Abr. de 2014.
[7] Thomas Hoblger y Norbert Jakowski. «Atmosferic Signal Propagation». En: *Springer Handbook of Global Navigation Satellite Systems*. Ed. por Peter J.G. Teunissen y Oliver Montenbruck. 2017. Cap. 19.
[8] Open Source Initiative. *The 2-Clause BSD License*. 2020. URL: `https://opensource.org/licenses/BSD-2-Clause`.
[9] Elliott D. Kaplan. «Introduction». En: *Understanding GPS/GNSS Principles and Applications*. Ed. por Elliot D. Kaplan y Christopher J. Hegarty. third. Artech House, 2017. Cap. 1.
[10] J. Liu y col. «Review of GNSS ambiguity validation theory». En: *Geo-mat. Inform. Sci.* (2014).
[11] C. Loosley y F. Douglas. *High-Performance Client/Server*. Wiley, 1997.
[12] Y Lou y col. «An algorithm and results analysis for GPS+ BDS inter-system mix double-difference RTK». En: *Geodesy Geodyn* (2016).
[13] Instituto Geográfico Nacional. *Red Argentina de Monitoreo Satelital Continuo*. 2020. URL: `http://www.ign.gob.ar/NuestrasActividades/Geodesia/Ramsac`.
[14] Dennis Odijk. «Positioning Model». En: *Springer Handbook of Global Navigation Satellite Systems*. Ed. por Peter J.G. Teunissen y Oliver Montenbruck. 2017. Cap. 19.
[15] Dennis Odijk y Lambert Wanninger. «Differential Positioning». En: *Springer Handbook of Global Navigation Satellite Systems*. Ed. por Peter J.G. Teunissen y Oliver Montenbruck. 2017. Cap. 26.
[16] Fan Ouyang y col. «Automatic delivery and recovery system of Wireless Sensor Networks (WSN) nodes based on UAV for agricultural applications». En: *Computers and Electronics in Agriculture* 162 (2019), págs. 31-43.
[17] Brent A. Renfro y col. *An Analysis of Global Positioning System (GPS) Standard Positioning Service Performance for 2019*. Inf. téc. The University of Texas at Austin, 14 de mayo de 2020.
[18] Rosendo Romero-Andrade y col. «Comparative analysis of precise point positioning processing technique with GPS low-cost in different technologies with academic software». En: *Measurement* 136 (2019), págs. 337-344.
[19] Tomoji Takasu. *RTKLIB: An Open Source Program Package for GNSS Positioning*. 2020. URL: `http://www.rtklib.com/`.
[20] Tomoji Takasu y Akio Yasuda. «Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB». En: *International Symposium on GPS/GNSS* (ene. de 2009).
[21] S. Chandra Yadav y S. Kumar Singh. *An Introduction to Client Server Computing*. New Age International Pvt. Ltd., Publishers, feb. de 2009.
[22] Yun Zhang y col. «Static and kinematic positioning performance of a low-cost real-time kinematic navigation system module». En: *Advances in Space Research* 63.9 (2019). Multi-GNSS: Methods, Benefits, Challenges, and Geosciences Applications, págs. 3029-3042.