E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Implementación de una base en un sistema de corrección RTK

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Iker Artajo Navarrete

Rafael Rogelio García Santos

Pamplona, 31 de mayo de 2023

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

Debido a la necesidad de mejorar la eficiencia y reducir costes en la producción agrícola, en los últimos años se ha comenzado a implementar técnicas como la agricultura autónoma con ayuda de sistemas de navegación por satélite. No obstante, a causa de distintos factores que distorsionan la señal satelital, la precisión de los sistemas de navegación se sitúa en el orden del metro, lo que obliga al empleo de técnicas de corrección de posicionamiento en tiempo real.

Para lograr esto, es necesario establecer un receptor satelital estático denominado “base” en un punto de referencia conocido que transmita el flujo de corrección que determine la desviación de la señal. Si se desea implementar una base, se debe determinar la posición en la que se sitúa mediante técnicas avanzadas de posicionamiento y transmitir el flujo de corrección, ya sea por radiofrecuencia o a través de Internet utilizando el protocolo de transmisión NTRIP.

En la siguiente memoria, se van a detallar los requerimientos necesarios para establecer una base operativa capaz de emitir un flujo de corrección al sistema de navegación de un vehículo agrícola autónomo.

Palabras clave

- GNSS
- RTK
- NTRIP
- Base
- Caster

Laburpena

Azken urteetan, nekazaritza produkzioaren efizientzia hobetzeko eta kostuak murrizteko beharrez, nabigazio sistemen laguntzarekin nekazaritza autonomoa inplementatu da. Hainbat arrazoi ezberdinengatik, nabigazio sistema hauek erabiltzen duten seinalea eraldatzen denez, metro mailako zehaztasuna lortzen da. Beraz, denbora errealean posizioa zuzentzeko teknikak nahitaezkoak dira.

Hau lortzeko, erreferentzia puntu ezagun batean, “base” izeneko estazio estatiko bat ezartzea ezinbestekoa da, seinalearen desbiderapenak zehazten dituen zuzenketak bidaltzen dituena. Base bat ezarri nahi bada, posizio teknika aurreratuak erabiliz, honen posizioa eta erabiliko den bidea zehaztu behar dira, irrati-maiztasun bidez edo NTRIP protokoloaren laguntzarekin Internet bidez.

Hurrengo orrialdeetan, ibilgailu autonomo bati zuzenketa fluxua bidaltzeko gai den base bat ezartzeko beharrezkoak diren eskakizunak azalduko dira.

Hitz-gakoak

- GNSS
- RTK
- NTRIP
- Base
- Caster

Abstract

Due to the need to improve efficiency and reduce costs in agricultural production, in recent years autonomous agriculture techniques have been implemented with the help of satellite navigation systems. However, due to several factors that distort the satellite signal, the precision of the navigation systems is in the order of meters, which requires the use of real-time positioning correction techniques.

To achieve this, it is necessary to establish a static satellite receiver known as a "base" at a known reference point, which transmits the correction flow that determines the signal deviation. If a base is to be implemented, its position must be determined using advanced positioning techniques, and the correction flow must be transmitted either via radio frequency or through the internet using the NTRIP data transmission protocol.

In the following report, the necessary requirements to establish an operational base capable of transmitting a correction stream to the navigation system of an autonomous agricultural vehicle will be detailed.

Keywords

- GNSS
- RTK
- NTRIP
- Base
- Caster

Índice

1 Introducción 1
1.1 Objeto del proyecto 1
1.2 Alcance 2
1.3 Antecedentes 2
2 Sistema Global de Navegación por Satélite (GNSS) 4
3 Sistema de corrección, Real-Time Kinematic (RTK) 7
3.1 Formato de datos 7
3.2 Protocolo de transmisión 8
3.2.1 Radio 8
3.2.2 NTRIP 9
4 Material y métodos 11
4.1 Posicionamiento de la base 11
4.1.1 Componentes 12
4.1.2 Configuración 13
4.2 Transmisión por Internet 18
4.2.1 Transmisión con ordenador 18
4.2.2 Transmisión por wifi 22
4.3 Establecimiento de un caster propio 26
4.3.1 Ejecución del Docker 27
4.3.2 Amazon Web Services (AWS) 29
4.4 Transmisión por radio 32
4.4.1 Componentes 32
4.4.2 Configuración 33

5 Resultados y discusión 34
5.1 Posicionamiento de la base 34
5.2 Transmisión por Internet 39
5.2.1 Transmisión con ordenador 39
5.2.2 Transmisión por wifi 41
5.3 Establecimiento de un caster propio 43
5.4 Transmisión por radio 46
5.5 Precisión posición del vehículo 47
6 Líneas futuras 50
7 Conclusión 51
8 Bibliografía 52
9 Agradecimientos 54

Índice de figuras

Figura 1. Recolección de arándano en Horticina. [2] 2
Figura 2. Trilateración con tres satélites en una superficie plana. [3] 4
Figura 3. Trilateración de cuatro satélites en sistema tridimensional. [3] 5
Figura 4. Esquema NTRIP. [7] 9
Figura 5. SimpleRTK2B. 12
Figura 6. Antena ANN-MB-00-00. 13
Figura 7. Habilitación de mensaje RXM-RAWX para USB. 14
Figura 8. Gráfica error en la posición respecto al tiempo. [9] 15
Figura 9. Conversión archivo UBX a RINEX. 16
Figura 10. Coordenadas de la antena 17
Figura 11. Modo fijo en U-Center. 18
Figura 12. Habilitación de mensajes RTCM. 20
Figura 13. Configuración puerto USB. 21
Figura 14. ESP32-C3-DevKit-02. 23
Figura 15. Configuración del UART1. 24
Figura 16. URLs a introducir en Arduino IDE. 25
Figura 17. Instrucción en la que se establece la posición de la base. 26
Figura 18. Archivos que componen la estructura de Docker. [13] 27
Figura 19. Instrucción para ejecutar el caster. 28
Figura 20. Esquema de servicios de AWS. 29
Figura 21. Reglas de entrada del grupo de seguridad. 30
Figura 22. Comando para iniciar sesión en la instancia creada en AWS. 30
Figura 23. Repositorio público para almacenar la imagen del Docker. 31
Figura 24. XBee SX868. 32
Figura 25. Adaptador para la conexión del XBee por USB. 32
Figura 26. Medición de 11 horas con la antena ANN-MB-00-00. 34
Figura 27. Medición de 15 horas con la antena ANN-MB-00-00. 35
Figura 28. Medición de 7 horas con la antena AS-ANT2B-SUR-L1L2-25SMA-00. 36
Figura 29. Medición de 13 horas con la antena AS-ANT2B-SUR-L1L2-25SMA-00. 37

Figura 30. Medición de 24 horas en un entorno abierto.....38
Figura 31. Resumen de la comunicación de la base con el caster en RTK2GO. 39
Figura 32. Simulación de cliente de la transmisión por ordenador.....40
Figura 33. Resumen de la comunicación con el caster en RTK2GO con I2C.....41
Figura 34. Resumen de la comunicación con el caster en RTK2GO con Serial.....42
Figura 35. Simulación de un cliente con una Base con protocolo Serial. 43
Figura 36. Configuración de SNIP para conectar con el caster que se ejecuta en Docker. 44
Figura 37. Archivos de registro del Docker ejecutado en localhost 45
Figura 38. Ancho de banda y número de clientes y bases conectados al caster. 45
Figura 39. Vehículo de Guiado Automático. 47
Figura 40. Cinta magnética que marca el recorrido que realiza el AGV. 47
Figura 41. Medición del vehículo con la antena ANN-MB-00-00. 48
Figura 42. Medición de la posición con la antena AS-ANT2B-SUR-L1L2-25SMA-00. 49

1 Introducción

La disminución de la población rural y la precariedad de las condiciones de trabajo han ocasionado la migración de los trabajadores agrícolas a otros sectores económicos. Por si no fuera suficiente, el constante aumento de la población mundial y una mayor conciencia en la sostenibilidad generan la necesidad de mejorar la eficiencia en la producción agrícola.

Para abordar esta situación, en los últimos años, se ha impulsado el uso de nuevas tecnologías en la producción agrícola. Por lo que cada vez es más recurrente recopilar información detallada de los cultivos mediante sensores, drones o software de análisis de datos que permiten maximizar el rendimiento y reducir los costes.

Entre las diversas técnicas empleadas, una que ha comenzado a popularizarse es la agricultura autónoma en la que se realizan tareas agrícolas sin la intervención humana directa. Para ello, se debe determinar la posición del vehículo mediante el empleo de sistemas de posicionamiento. Sin embargo, al ser necesario conocer la posición de manera precisa, los sistemas de posicionamiento empleados requieren de técnicas de corrección de posicionamiento en tiempo real los cuales precisan de una base estática en un punto de referencia conocido.

1.1 Objeto del proyecto

En la presente memoria, se van a detallar los requerimientos necesarios para establecer una base operativa capaz de emitir un flujo de corrección al sistema de navegación de un vehículo agrónomo autónomo. Con el fin de cumplir con el propósito planteado, se establecen los siguientes objetivos:

- Hallar las coordenadas de la ubicación donde se va a estacionar la base con la máxima precisión posible.
- Transferir el flujo de corrección que proporciona la base por radiofrecuencia e Internet con ayuda del protocolo NTRIP para poder asegurar la mejor señal en cada momento.
- Establecer un caster de NTRIP propio para tener una mayor seguridad y fiabilidad de los datos enviados por este protocolo.

1.2 Alcance

Este proyecto está planteado para que se pueda establecer una base operativa en la Finca Horticina situada en Cerdido, Galicia. Se trata de la mayor plantación de arándanos del norte peninsular con 58’5 ha en el año 2018.

En el año 2020, Horticina fue adquirida por Cool Berries una empresa con una larga trayectoria en el negocio agrícola, sostenible y tecnológico con el objetivo de cambiar el negocio tradicional del arándano. Desde entonces, se ha aumentado el terreno de plantación hasta alcanzar las 82 ha con las que cuenta hoy en día [1].

1.3 Antecedentes

Debido a la gran extensión del terreno eran necesarios más de 400 trabajadores durante más de dos meses para recolectar manualmente los arándanos del terreno. Con el reciente cambio de estrategia de la compañía, causada principalmente por el Brexit al ser Reino Unido su principal comprador, se decidió comenzar a emplear cosechadoras para la recolección mecánica de los arándanos, tal y como se muestra en la Figura 1. De esta forma, se logró disminuir hasta 25 los trabajadores necesarios, reduciendo el coste en la recolección en más de un 600 %.

Figura 1. Recolección de arándano en Horticina. [2]

El siguiente paso que se ha decidido tomar por parte de Cool Berries para volverse más eficientes, ha sido emplear tractores eléctricos autónomos para arrastrar las cosechadoras. Para ello, se ha optado por instalar un sistema fotovoltaico aislado que permita alimentar tanto a estos vehículos como a las instalaciones colindantes a la zona de cultivo.

Para poder llevar a cabo el proyecto, se ha decidido realizar una colaboración con IED, empresa en la que el redactor del informe está cursando las prácticas curriculares propias del grado de Ingeniería en Tecnologías Industriales. IED ha creado la sección Mula para el desarrollo de este proyecto, en el que en el acuerdo establecido se ha designado como la entidad responsable del control del vehículo.

2 Sistema Global de Navegación por Satélite (GNSS)

El Sistema Global de Navegación por Satélite (GNSS) está formado por una red de satélites artificiales que permiten determinar la posición tridimensional en tiempo real en cualquier lugar de la superficie terrestre. El sistema GNSS incluye distintos sistemas de navegación por satélite, siendo los más destacados GLONASS, Galileo, BeiDou y el popularmente conocido GPS.

Los satélites orbitan alrededor de la tierra emitiendo señales codificadas con la hora atómica en la que la señal se difunde y el receptor tiene conocimiento de la hora atómica en la que la señal es recibida. Por lo tanto, se conoce el tiempo que tarda la señal en recorrer la distancia desde el satélite hasta el cuerpo. Sabiendo además que la señal viaja aproximadamente a la velocidad de la luz, se puede precisar la distancia entre los dos puntos. De modo que, calculando la distancia respecto a varios satélites, mediante esta técnica de posicionamiento conocida como trilateración, se puede determinar la posición del receptor. En la Figura 2**,** se puede ver un ejemplo de este método en una superficie plana donde al emplear tres puntos de referencia dejan una única ubicación posible para el cuerpo.

Figura 2. Trilateración con tres satélites en una superficie plana. [3]

Sin embargo, en un sistema tridimensional la posible ubicación del receptor se ubica en la superficie de una esfera en cuyo centro se encuentra el satélite. De forma que, al medir la distancia con 3 satélites, se crearían dos puntos de intersección distintos: un punto con la ubicación del receptor y otro fuera de la superficie terrestre. Por lo tanto, para poder determinar la latitud, longitud y altitud del receptor en un sistema tridimensional es necesario disponer de al menos 4 satélites. El receptor GNSS se encontraría en el punto de intersección de todas las esferas, como se puede ver en la Figura 3.

Figura 3. Trilateración de cuatro satélites en sistema tridimensional. [3]

Sin embargo, el sistema GNSS posee distintos errores de medida que por pequeños que sean, teniendo en cuenta la velocidad a la que viaja la señal, causan un error de posición considerable. La causa de error más destacada es debido al efecto atmosférico en la propagación de la señal, especialmente por las perturbaciones en la ionosfera y en la troposfera. La ionosfera es una región de la atmósfera que contiene un elevado número de iones, cuanto más ionizada se encuentre, mayores son las perturbaciones que origina. Mientras que en la troposfera se producen errores debido a los fenómenos meteorológicos, al estar nuboso o llover hay una mayor concentración de vapor de agua que aumenta la reflexión y, por tanto, mayores son los errores en la medida.

A esto se le debe sumar otras causas de menor peso, como la desviación en los relojes de los receptores y de los satélites, el efecto *multipath* causada por la reflexión de la señal o el ruido, que hacen que la exactitud de este sistema no sea menor de 1 o 2 m.

Por lo tanto, aplicaciones que requieran de un margen de maniobra estrecho, se ven en la obligación de emplear técnicas que reduzcan el error. En el caso en el que se centra está memoria, se emplea la técnica *Real-Time Kinematic* (RTK) que se explicará en el siguiente apartado y que corrige las perturbaciones de la ionosfera y troposfera para lograr una precisión en tiempo real del orden del centímetro.

3 Sistema de corrección, Real-Time Kinematic (RTK)

El sistema georreferenciado de datos *Real-Time Kinematic* (RTK) o también conocido como *Differential GPS* (DGPS) tiene como objetivo reducir el error de posición producido con el sistema GNSS mediante el envío de un flujo de corrección. Para ello, se emplean dos receptores GNSS, uno de ellos estático denominado base y el otro móvil conocido como rover o vehículo. La estación base se coloca en un punto de referencia con coordenadas conocidas, de forma que, al recibir la señal del satélite, es capaz de determinar la distorsión de las señales y trasmitir las correcciones al receptor móvil.

De esta manera, se establece la posición en tiempo real del rover con una precisión de menos de 2 cm a pocos kilómetros de la base. La precisión de la posición del receptor móvil va en función de la distancia entre los dos receptores, aumentando el error de la posición en 2 cm por cada 10 km entre la base y el vehículo o rover [4].

A la hora de medir la posición de manera precisa, también es posible emplear el sistema *Post-Processed Kinematic* (PPK) que al igual que el sistema RTK requiere de dos receptores GNSS para poder trabajar. No obstante, a diferencia del RTK, el sistema PPK no obtiene la solución en tiempo real, evitando de esta forma el riesgo de pérdida de precisión debido a una interrupción en el enlace de datos.

En el caso en el que se centra esta memoria, es fundamental que el vehículo circule por la ruta estipulada, por lo que dentro de las técnicas citadas solo es posible trabajar con el sistema RTK, ya que con el sistema PPK no es posible realizar correcciones en tiempo real.

3.1 Formato de datos

Los datos que envía la base a la estación móvil para la transmisión de datos en tiempo real pueden tener distintos formatos, pudiéndose dividir en formato propietario y en RTCM SC-104.

El formato propietario, también conocido como *raw-data*, son los protocolos desarrollados por los fabricantes de cada equipo y suelen ser usados por equipos de la misma marca comercial. En caso de desear establecer comunicación entre dispositivos de distintas marcas, es necesario emplear otros estándares.

Radio Technical Comission for Maritime (RTCM) es un organismo que ideó el RTCM SC-104 con el fin de comunicar en formato binario las posiciones de los barcos de Estados Unidos. Actualmente, se considera como estándar internacional para indicar los bytes de datos de corrección que permiten calcular la posición en los sistemas diferenciales de navegación global por satélite. Dentro del formato RTCM SC-104 se pueden diferenciar la versión 2.x y la versión 3.x. La versión 3.x es la más actualizada, favoreciendo las operaciones RTK al emplear un menor ancho de banda, sin embargo, dispositivos antiguos no están adaptados y deben usar la versión 2.x.

3.2 Protocolo de transmisión

Se pueden emplear diversos medios para enviar el flujo de corrección calculado por la estación de referencia al receptor móvil, ya sea por radiofrecuencia, wifi o *bluetooth*. En el caso en el que se centra este informe, no se va a hacer uso de la comunicación *bluetooth*, al trabajar con distancias de un mayor orden que con las que se comunica este protocolo. Por lo tanto, se van a emplear tanto la radiofrecuencia y como el wifi, comparándolas para poder tomar la mejor señal en cada momento.

3.2.1 Radio

El envío de señales a través de radio es un proceso en el que se transmite señales de información por medio de ondas de radio que se propagan por el espacio libre. Para las transmisiones de correcciones RTK se utilizan radios *Ultra High Frequency* (UHF) con una banda de frecuencia con un rango entre 300 MHz y 3GHz o *Very High Frequency* (VHF) entre 30 y 300 MHz. Es más común utilizar radios UHF ya que ofrecen una mayor potencia de transmisión y un alcance más amplio.

Este tipo de transmisión tiene una limitación en el alcance de unos 20 kilómetros [5], por lo que es necesario emplear repetidores en caso de desear mejorar la cobertura. Además, puede haber interrupciones por obstáculos en el terreno, por lo que es importante que el terreno sea lo más llano posible.

No obstante, a favor de este tipo de comunicación hay que destacar que no depende de la disponibilidad de una red celular. Por tanto, es una comunicación estable que no depende de la calidad de la señal de Internet ni la congestión de la red. Además, las transmisiones por radio suelen ser menos vulnerables a la interceptación de datos que otras transmisiones.

3.2.2 NTRIP

El transporte del flujo de corrección emitido por la base en formato RTCM a un receptor móvil a través de Internet es conocido como protocolo *Network Transportation of RTCM via Internet Protocol* (NTRIP). Este protocolo fue desarrollado en el año 2002 por la Agencia Federal de Cartografía y Geodesia Alemana (BKG) con el fin de transmitir las correcciones diferenciales en formato RTCM por medio del protocolo HTTP empleado en la Web [6].

Tal y como se expresa en el esquema de la Figura 4, para poner en marcha el sistema NTRIP se requieren tres elementos: servidor NTRIP, caster NTRIP y cliente NTRIP

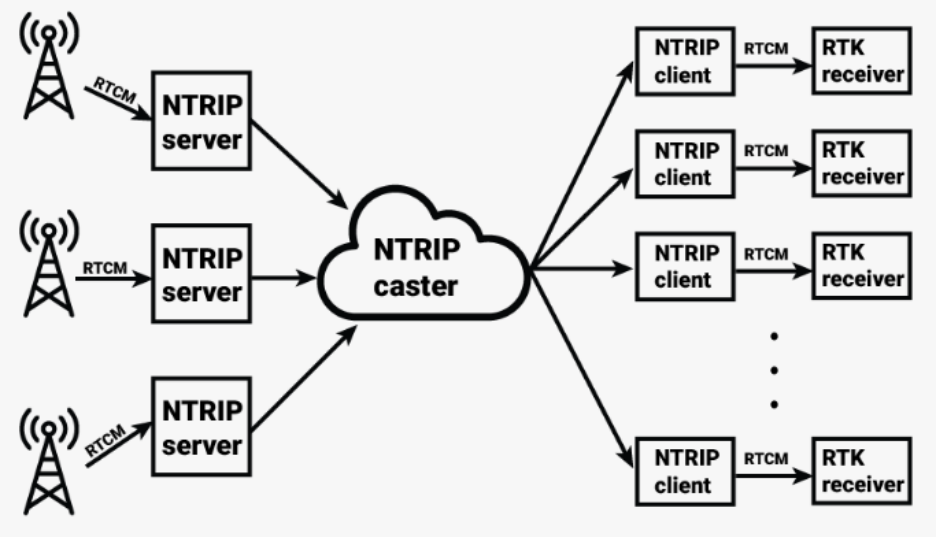


Figura 4. Esquema NTRIP. [7]

El servidor transmite las correcciones RTK al caster para que pueda distribuirlas a los clientes, para ello suele utilizarse la propia base como servidor. Es habitual emplear estaciones permanentes de servicio de posicionamiento públicos como la Red de Geodesia Activa de Navarra (RGAN) o el Instituto Geográfico Nacional (IGN) de los que se obtiene la corrección de manera gratuita. Sin embargo, al no tener a disposición ninguna estación permanente en las proximidades de la zona de trabajo es necesario montar una base propia.

El caster es el servicio que administra el flujo de datos proveniente de una o varias estaciones base y las distribuye a los clientes. Al igual que con el servidor, es posible crear un caster propio, aunque cabe recalcar que actualmente existe un gran número de caster gratuitos como RTK2GO o Emlid.

El cliente es el receptor RTK que recibe las correcciones del caster, siendo por tanto los receptores GNSS que actúan como rover.

A diferencia de la comunicación por radio en el que la comunicación tiene un alcance limitado, al utilizar NTRIP no hay una limitación en el rango de comunicación. Además, se evitan las posibles interferencias de radio causadas por obstáculos en el terreno, por lo que es mejor alternativa en entornos urbanos. No obstante, pese a ser una comunicación normalmente fiable hay que tener en cuenta que para que la comunicación NTRIP sea posible es necesario que tanto la base como el rover tengan acceso a Internet.

4 Material y métodos

En el siguiente apartado, se explican de manera detallada los componentes y pasos a realizar para poder lograr una transmisión de la base con una precisión suficiente como para controlar de manera autónoma un vehículo agrícola.

Como se ha explicado en la introducción, la base es la responsable de transmitir al vehículo el flujo que permite corregir su posición. Por razones tanto económicas como temporales, se recomienda comprobar en el IGN si se dispone de alguna base activa con la que poder trabajar. En el caso en el que se centra la memoria, al actuar en un entorno rural no se tiene a disposición ninguna estación permanente en las inmediaciones, por lo que se debe armar una base propia.

4.1 Posicionamiento de la base

Para que la base sea capaz de enviar la corrección es necesario conocer la posición de la base de manera muy precisa. En caso de no conocer las coordenadas de la ubicación es posible obtenerla mediante la técnica de posicionamiento *Precise Point Positioning* (PPP). Este sistema de posicionamiento avanzado modela los errores del sistema GNSS desde un único receptor, midiendo la diferencia en la fase de la señal en dos frecuencias distintas. La fase de la señal es proporcional a la distancia que recorre y es inversamente proporcional a la longitud de onda, tal y como se expresa en la siguiente ecuación.

$$\Delta\varphi = 2\pi \frac{d}{\lambda} \qquad \text{(ec. 1)}$$

La longitud de onda de cada una de las frecuencias es diferente, ya que, aunque se suponga que viajan a la misma velocidad, en este caso a la velocidad de la luz, difieren en la frecuencia. Por lo tanto, la señal con mayor frecuencia tiene una menor longitud de onda, como demuestra la próxima ecuación.

$$\lambda = \frac{c}{f} \qquad \text{(ec. 2)}$$

De forma que partiendo de la misma fase inicial y recorriendo la misma distancia, la diferencia de fase entre las dos frecuencias es proporcional a la distancia entre receptor y satélite y a la velocidad a la que circulan.

$$\Delta\varphi = 2\pi d\frac{c}{f_1} - 2\pi d\frac{c}{f_0} \quad \text{(ec. 3)}$$

De manera que la distancia entre receptor y satélite se puede calcular mediante la siguiente expresión.

$$d = \frac{\Delta\varphi\ c}{2\pi\ \Delta f} \quad \text{(ec. 4)}$$

4.1.1 Componentes

A pesar de que en futuros capítulos se precise de un mayor número de dispositivos, para poder determinar la posición de la base tan solo son necesarios una placa de desarrollo que incluya un módulo de posicionamiento, una antena compatible y un sistema con U-Center.

Como placa de desarrollo se ha optado por la simpleRTK2B que puede verse en la Figura 5, basada en el módulo de GNSS de alta precisión ZED-F9P. Este dispositivo es capaz de realizar correcciones de señal diferencial en tiempo real con múltiples constelaciones, entre las que se incluyen GPS, GLONASS, Galileo y BeiDou, lo que le permite proporcionar información precisa de posicionamiento, navegación y tiempo.

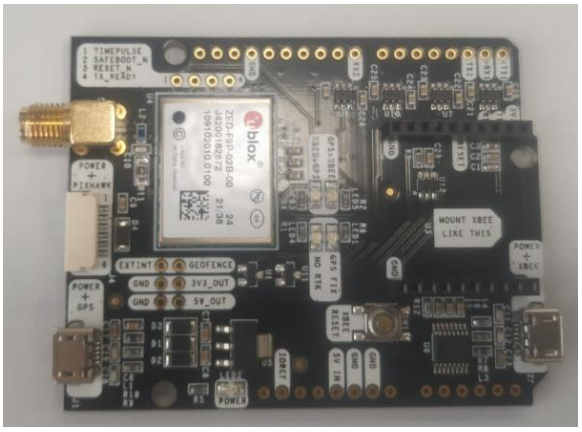


Figura 5. SimpleRTK2B.

Para que el sistema pueda comunicarse es necesario enlazar por conexión SMA una antena de la misma marca que el receptor GNSS, en este caso la marca suiza U-blox. Por ello, se ha escogido la antena ANN-MB-00-00 de la Figura 6, una antena capaz de operar en múltiples frecuencias para cada constelación de satélites con las que se trabajan.

Figura 6. Antena ANN-MB-00-00.

Finalmente, para establecer la posición de la base es necesario contar con un ordenador que recopile los datos GNSS y le suministre alimentación a la base. Para lograr esto, se conecta el ordenador al puerto Power+GPS del SimpleRTK2B con un conector de micro-USB y se emplea U-center[1], un software de evaluación GNSS de la compañía U-blox, para configurar la base.

4.1.2 Configuración

Para lograr las coordenadas de la ubicación, se debe posicionar la antena de la base en una ubicación fija, recopilar datos GNSS sin procesar y enviarlos a un centro de procesamiento para PPP, de forma que se pueda configurar el receptor en modo fijo.

A la hora de escoger emplazamiento es fundamental optar por una superficie fija con una visión clara del cielo, sin ningún tipo de obstáculo que pueda interferir en la medición, siendo lo más habitual situarlo en un tejado.

[1] Enlace de descarga en el anexo.

Una vez que se tenga la ubicación permanente de la antena, se emplea U-Center con el fin de emitir los datos sin procesar. Tras conectar el SimpleRTK2B al ordenador y seleccionar el correspondiente COM se debe comprobar que el firmware se encuentre en la última versión (HPG 1.32) desde *Tools -> Firmware Update.* A continuación, se configura el receptor para que emita datos sin procesar de los satélites, para ello se selecciona el modo encuesta (survey in) desde *View -> Configuration View -> TMODE3* con Minimum Observation time de 180 s y Required Position Accuracy de 1 m y se habilita el mensaje RXM-RAWX para USB en MSG como se muestra en la Figura 7. Tras guardar la configuración puede tardar unos minutos en comenzar a recibir señal en el U-Center, momento en el que pasa de “No Fix” a “3D/DGNSS”.

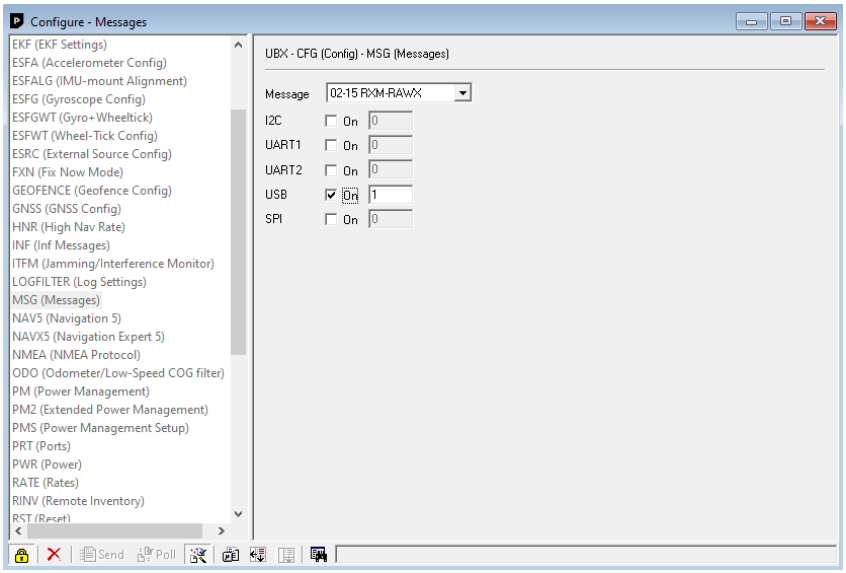


Figura 7. Habilitación de mensaje RXM-RAWX para USB.

Al pulsar el botón de grabación, todos los datos del receptor se almacenan en un archivo de extensión UBX. Sabiendo que el receptor se encuentra en una posición fija, se puede reducir el error calculando repetidamente su posición y promediando los resultados, de modo que como se puede ver en la Figura 8, el error de posición de la base es menor cuanto mayor sea el tiempo de grabación. Se recomienda tomar una muestra de al menos 6 horas y a ser posible de 24 horas para establecer la ubicación de la base [8].

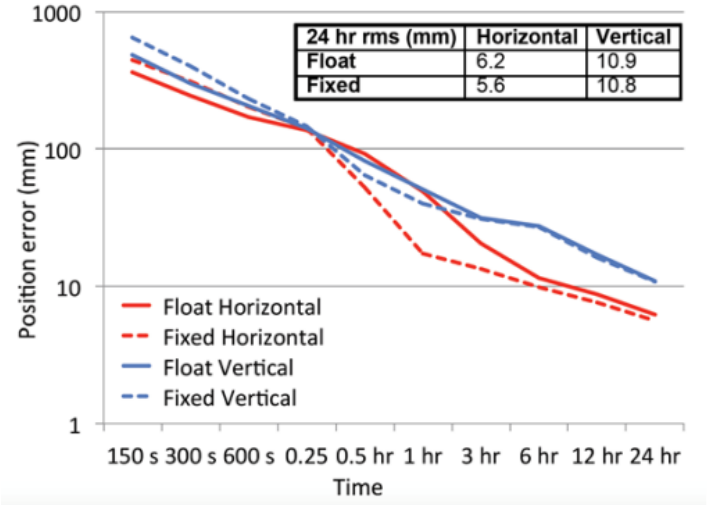


Figura 8. Gráfica error en la posición respecto al tiempo. [9]

Una vez se haya logrado la muestra pertinente, se debe convertir el archivo UBX a RINEX, el formato de archivo estándar utilizado para almacenar e intercambiar datos GNSS, empleando RTKLIB[2]. Para realizar la conversión, se abre RTKCONV mostrado en la Figura 9, se selecciona el archivo UBX y se presiona *convertir*. Tras unos segundos de espera se debería obtener un archivo OBS.

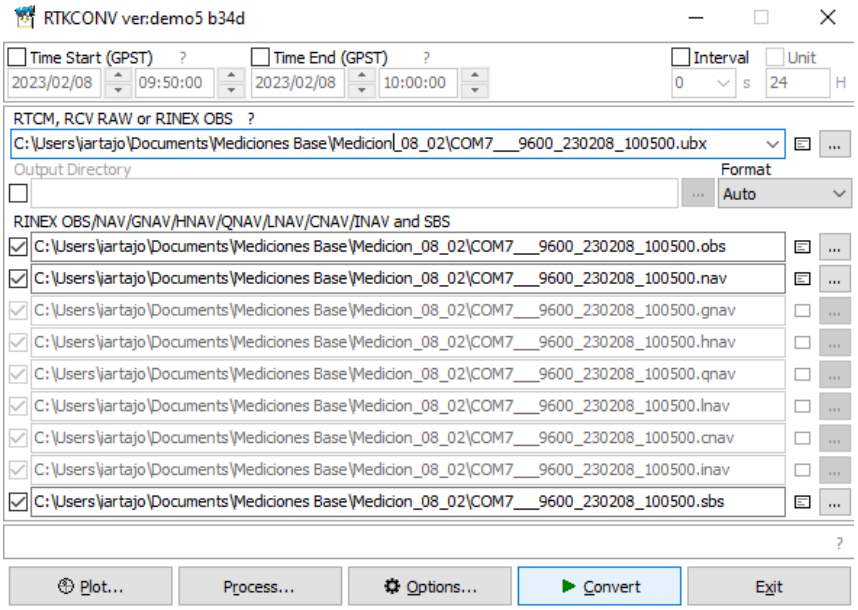
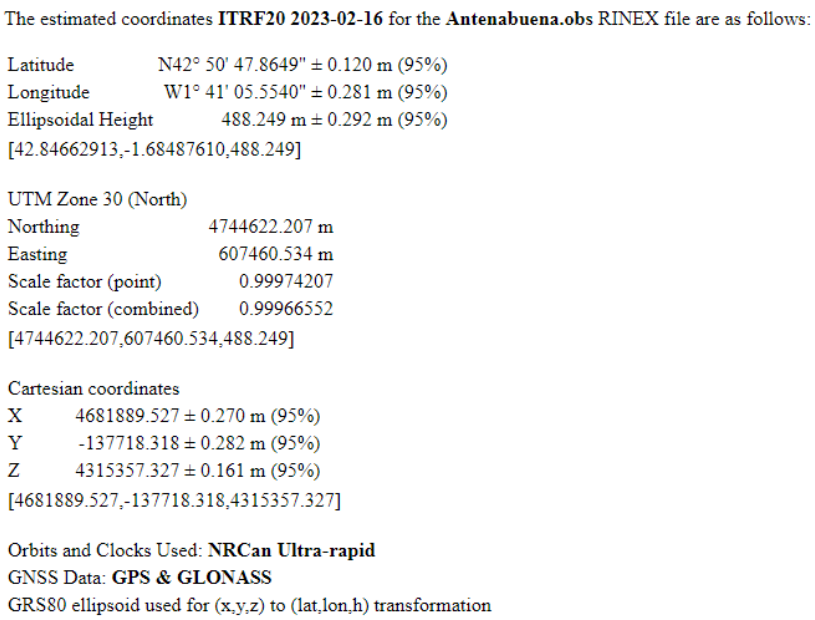


Figura 9. Conversión archivo UBX a RINEX.

[2] Enlace de descarga en el anexo.

Tras obtener los datos satelitales sin procesar en formato RINEX, deben enviarse a un centro de posprocesamiento para lograr la ubicación real de la antena. El centro de posprocesamiento procesa los datos de observación, aplicando algoritmos y modelos para eliminar errores causados por retrasos atmosféricos y errores de reloj. En este caso, se ha optado por el servicio canadiense CSRS-PPP, ya que es un servicio gratuito, disponible en todo el mundo y con una alta velocidad de procesamiento. Al enviar el archivo OBS comprimido, pasadas unas pocas horas se debería recibir una respuesta con los resultados en el correo con el que se creó la cuenta. En el resumen del correo recibido, se encuentran las coordenadas en formato cartesiano, UTM y geográficas de la base, tal y como se puede observar en la Figura 10.



The estimated coordinates **ITRF20 2023-02-16** for the **Antenabuena.obs** RINEX file are as follows:

Latitude N42° 50' 47.8649" ± 0.120 m (95%)
Longitude W1° 41' 05.5540" ± 0.281 m (95%)
Ellipsoidal Height 488.249 m ± 0.292 m (95%)
[42.84662913,-1.68487610,488.249]

UTM Zone 30 (North)
Northing 4744622.207 m
Easting 607460.534 m
Scale factor (point) 0.99974207
Scale factor (combined) 0.99966552
[4744622.207,607460.534,488.249]

Cartesian coordinates
X 4681889.527 ± 0.270 m (95%)
Y -137718.318 ± 0.282 m (95%)
Z 4315357.327 ± 0.161 m (95%)
[4681889.527,-137718.318,4315357.327]

Orbits and Clocks Used: **NRCan Ultra-rapid**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 10. Coordenadas de la antena

Finalmente, para establecer la posición de la base se vuelve a TMODE3, se selecciona el modo fijo y se introducen las coordenadas cartesianas obtenidas en el paso anterior, como se puede apreciar en la Figura 11. De este modo, el receptor GNSS comenzaría a actuar como base transmitiendo datos de corrección.

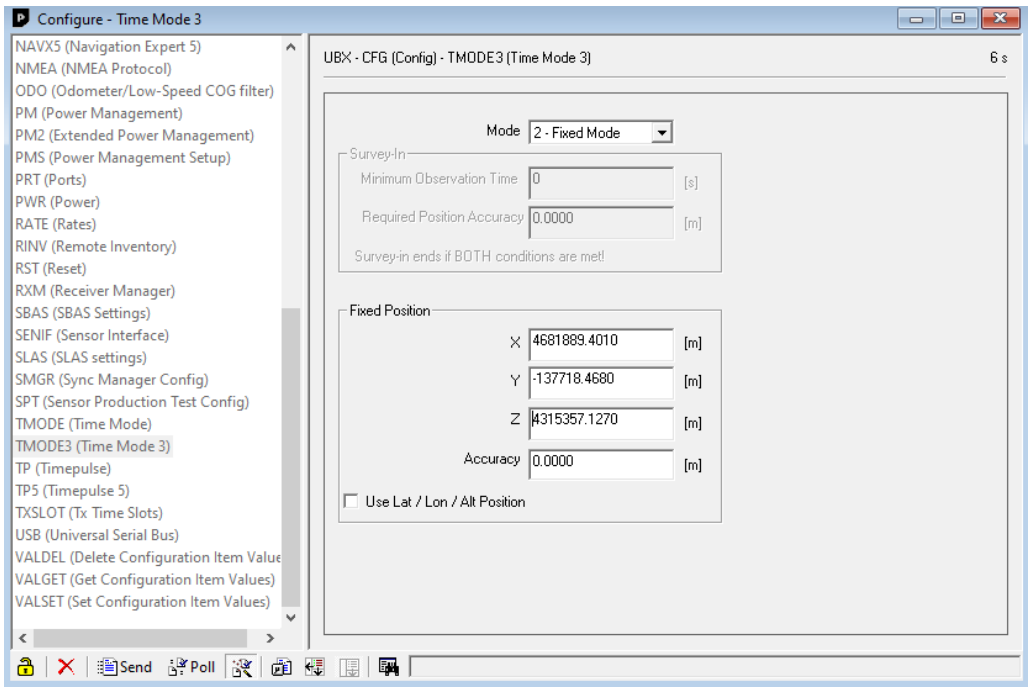


Figura 11. Modo fijo en U-Center.

4.2 Transmisión por Internet

Se pueden emplear diversos medios para la comunicación del flujo de corrección de la base, este punto se centra en la transmisión por Internet. Como ya se comentó anteriormente, la transmisión de datos de navegación satelital a través de Internet se designa con el acrónimo NTRIP. En el presente apartado, se detalla la configuración pertinente para la transmisión NTRIP tanto para una base conectada a un ordenador, como para una base con un ESP32.

4.2.1 Transmisión con ordenador

Es posible utilizar un ordenador que reciba los datos en serie de la base a través del puerto 2101 y los reenvíe a través de Internet al caster. De esta forma, se obtiene una comunicación fácilmente gestionable, pero que no tiene un inicio automático en caso de pérdidas de energía.

Se puede emplear un escritorio remoto en caso de que la base este en una ubicación de difícil acceso, para poder gestionar el ordenador al que va conectado la base desde otro dispositivo. Aunque en este informe no se va a dar información adicional sobre el escritorio remoto, al alejarse del tema en el que se centra el trabajo.

4.2.1.1 Componentes

Al igual que en el apartado 4.1, se requiere de un SimpleRTK2B, una antena ANN-MB-00-00 conectada por SMA y un ordenador enlazado por el puerto POWER+GPS para transmitir el flujo de corrección de la base. No obstante, a diferencia del posicionamiento de la base que tan solo necesita de U-center para configurar el receptor GNSS, también requiere de SNIP para transmitir la corrección al caster.

SNIP[3] es un software de estación base GNSS que permite distribuir correcciones de señal en tiempo real a través de Internet. Para poder trabajar con esta aplicación, es necesario completar un sencillo registro que permite acceder a una versión de prueba, que pasa a ser una versión gratuita completa, una vez confirmada la cuenta.

4.2.1.2 Configuración

Una vez se hayan establecido las coordenadas de la base, se debe configurar los mensajes que va a transmitir al caster a través de Internet. En *View -> Configuration View -> MSG,* se inhabilitan los mensajes sin procesar RXM-RAWX por USB que se grabaron en la recopilación de datos GNSS para determinar la posición de la base [10].

[3] Enlace de descarga en el anexo.

A continuación, se habilitan la comunicación por USB de los mensajes RTCM 3.3 que se establecen a continuación:

- RTCM 3.3 1005, que aporta información general de la estación base, como son la localización de la antena y la calidad de la señal recibida.
- RTCM 3.3 1074, el cual proporciona correcciones de fase de satélites GPS.
- RTCM 3.3 1084, concediendo correcciones de fase de satélites GLONASS.
- RTCM 3.3 1094, que suministra correcciones de fase de satélites Galileo.
- RTCM 3.3 1124, el cual aporta correcciones de fase de satélites BeiDou.
- RTCM 3.3 1230, que proporciona correcciones para los parámetros de órbita de los satélites GLONASS.

Estos mensajes se emiten a una frecuencia de 1 Hz, tal y como se muestra en la Figura 12, a excepción del mensaje 1230 que se transmite cada 10 segundos, ya que la naturaleza no se mueve lo suficientemente rápido como para requerir una frecuencia mayor [8]. Además, una frecuencia más alta produciría un aumento del flujo de información transmitido.

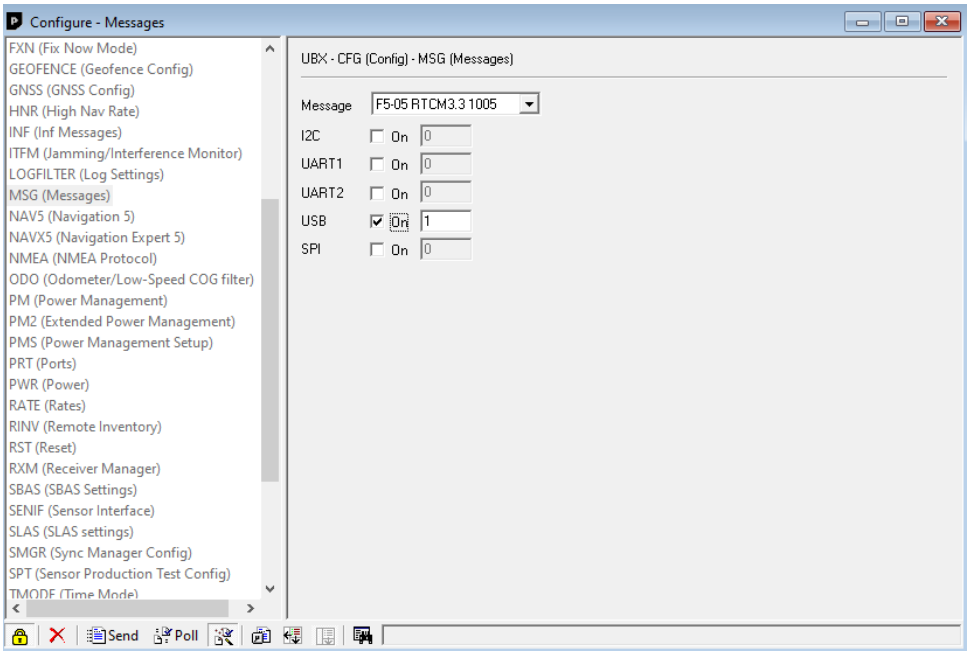


Figura 12. Habilitación de mensajes RTCM.

Los mensajes NMEA son una serie de mensaje de datos en formato ASCII utilizado en sistema de navegación por satélite para transmitir datos relevantes de la navegación. Sin embargo, a diferencia de los RTCM que están diseñados para la transmisión de correcciones diferenciales en tiempo real, los NMEA, al ser menos compactos, están diseñados para la transmisión de datos de baja velocidad. Aunque sea posible deshabilitar la emisión de todos los mensajes NMEA, también es posible dejar habilitados los siguientes tipos:

- NMEA GxGGA, que proporciona información de posición, tiempo y calidad de la señal satelital.
- NMEA GxGLL, este mensaje aporta información de posición en términos de latitud y longitud.
- NMEA GxGSA, el cual provee información sobre la geometría de los satélites activos y la precisión de la medición.
- NMEA GxGSV, que ofrece información sobre los satélites visibles y sus señales.

Para terminar con la configuración a través de U-center, se debe configurar el puerto USB desde *View -> Configuration View -> PRT,* asegurando que el protocolo tanto de entrada como de salida sea UBX+NMEA+RTCM3, como se aprecia en la Figura 13.

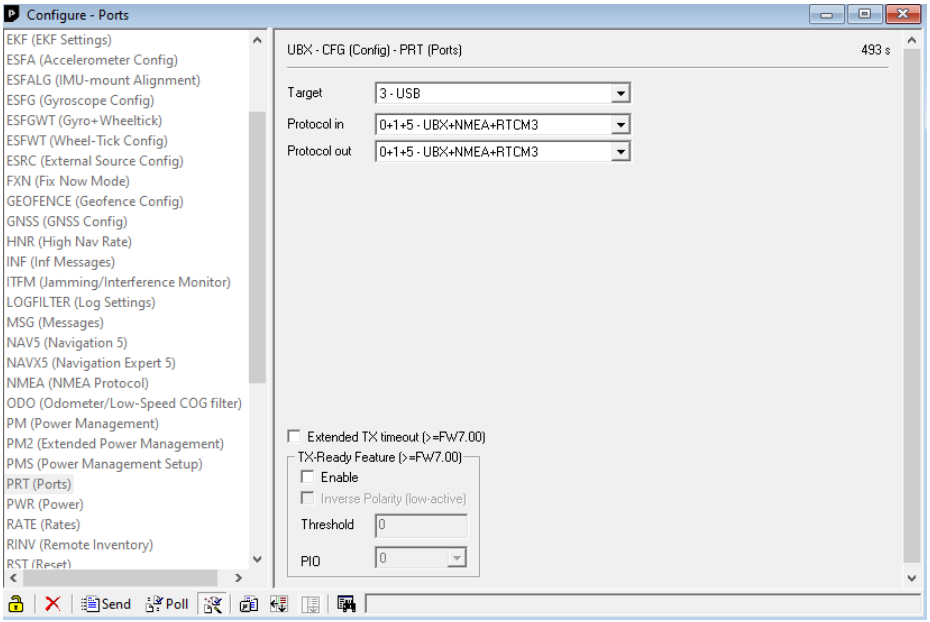


Figura 13. Configuración puerto USB.

Tras configurar la base para que emita el flujo de información al ordenador, es necesario enviarlo a un NTRIP caster con la ayuda de SNIP. En caso de utilizar el caster público RTK2GO, se debe crear un *mountpoint*[4] de manera previa. El *mountpoint* es un punto de acceso único para cada una de las estaciones base que permite a los clientes acceder a sus correcciones.

Ya dentro de SNIP, en el apartado *Serial Streams*, se selecciona *Add New Stream* y se introducen los datos de la comunicación con la base y la información del *mountpoint* (*baud rate,* COM, nombre, contraseña, ubicación, …). Al crear el nuevo stream, se debe conectar haciendo clic derecho en la nueva columna generada, de forma que se establece comunicación entre la base y el ordenador.

Finalmente, para enviar la corrección al caster, desde *Pushed-Out Streams,* se marca *Add Stream …* y se introducen el *mountpoint*, su contraseña, la url del caster al que se desea enviar la corrección y como puerto el 2101, estándar para la transmisión de datos a través del protocolo NTRIP. Al igual que al crear el stream, se debe conectar haciendo clic derecho, de forma que tras seguir los pasos proporcionados se establece la comunicación con el caster.

4.2.2 Transmisión por wifi

En caso de no desear emplear un ordenador para transmitir el flujo de corrección, es posible emitir la corrección por wifi a partir de un ESP32. De este modo, se logra un sistema con un consumo de energía menor que con un ordenador y menos susceptible a cambios causados por actualizaciones del sistema.

4.2.2.1 Componentes

Para poder transmitir al caster, es necesario que la base este formada por un ESP32 y por el SimpleRTK2B y la antena ANN-MB-00-00 ya citadas en el apartado 4.1.1. El ESP32 es un microcontrolador de bajo consumo diseñado para ser utilizado en una amplia variedad de aplicaciones de IoT.

[4] Enlace para poder crear el Mountpoint en el anexo.

Se va a hacer uso de la placa de desarrollo ESP32-C3-DevKit-02 que puede verse en la Figura 14, capaz de comunicarse por wifi y *bluetooth*, aunque como ya se ha comentado anteriormente, únicamente se va a hacer uso del wifi para transmitir el flujo de corrección generado por el SimpleRTK2B, debido a las distancias con las que se trabaja.

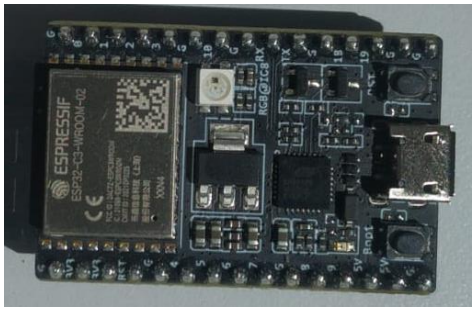


Figura 14. ESP32-C3-DevKit-02.

La conexión entre el ESP32 y el SimpleRTK2B puede realizarse tanto por el protocolo de comunicación Serial, como por el I2C en caso de tener un SimpleRTK2B de versión 3 o superior.

En la Tabla 1, se pueden ver las conexiones a realizar en cada uno de los protocolos. El conexionado Serial del SimpleRTK2B se realiza a través del Serial1 ubicado en la salida POWER+PIXHAWK, en donde el segundo terminal corresponde al RX y el tercer terminal al TX.

Tabla 1. Conexionado Serial e I2C entre SimpleRTK2B y ESP32.

Serial		
SimpleRTK2B		**ESP32**
5V	↔	5V Out
GND	↔	GND
RX	↔	3 (TX)
TX	↔	1 (RX)

I2C		
SimpleRTK2B		**ESP32**
5V	↔	5V Out
GND	↔	GND
SDA	↔	21
SCL	↔	22

Además, se requiere de un ordenador con Arduino IDE[5] desde el que se introduce la programación necesaria al ESP32 y con U-center para poder configurar la comunicación Serial del SimpleRTK2B.

4.2.2.2 Configuración

Tras introducir las coordenadas de la base como se explica en el apartado 4.2.1.2, se debe transmitir la información a través del UART1 del SimpleRTK2B. Para ello, como se ve en la Figura 15, desde U-center en *View -> Configuration View -> PRT* se ajusta el baud rate a 9600, garantizando que el protocolo tanto de entrada como de salida sea UBX+NMEA+RTCM3.

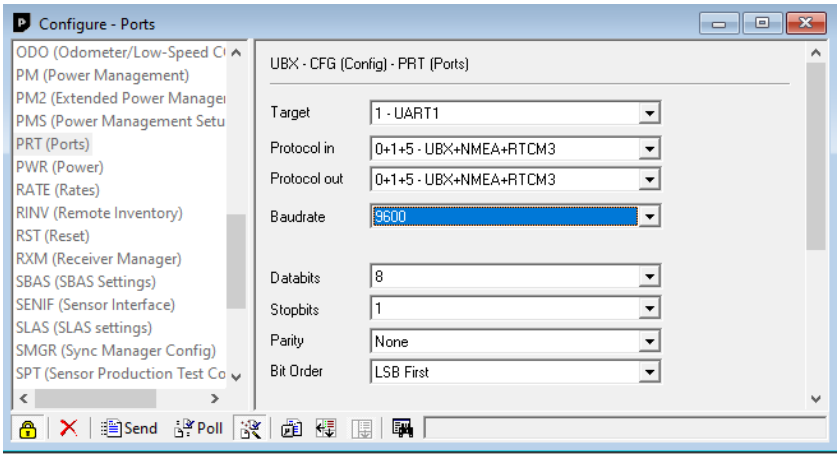


Figura 15. Configuración del UART1.

[5] Enlace de descarga en el anexo.

Una vez la comunicación del SimpleRTK2B este configurada es turno de programar el ESP32 desde Arduino IDE. Para poder trabajar desde Arduino con un ESP32 es necesario realizar algunas configuraciones previas. De modo que desde *Archivo -> Preferencias* en el *Gestor de URLs adicionales de Tarjetas* se introducen las URLs de la Figura 16.

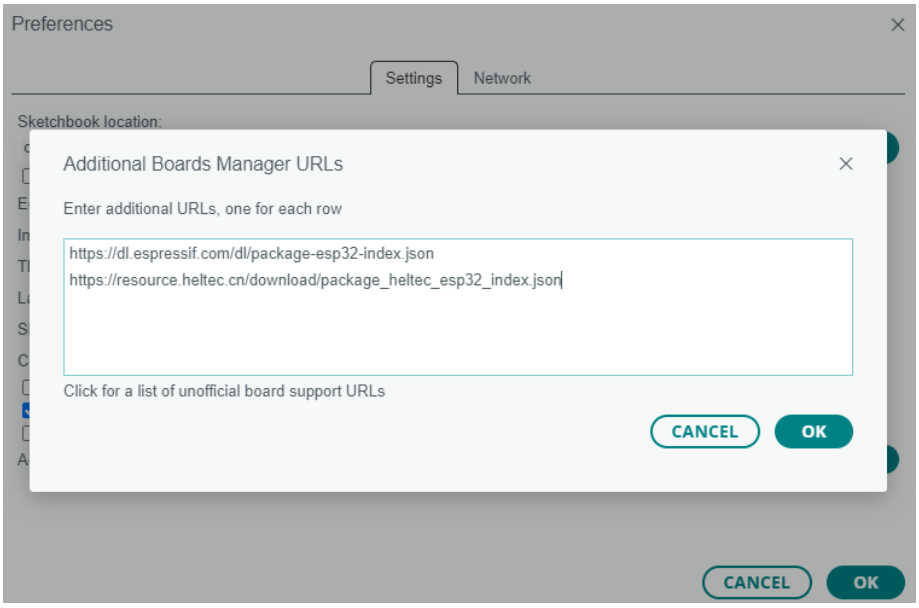


Figura 16. URLs a introducir en Arduino IDE.

A continuación, se instala el soporte para ESP32 desde *Herramientas -> Placas -> Gestor de tarjetas* y se comienza a actualizar la base de datos utilizando la URLs agregadas en el paso anterior.

Una vez programado el Arduino IDE para permitir trabajar con ESP32, se conecta el ESP32 al ordenador y se selecciona la placa y puerto correspondientes[6]. Seguidamente se introduce la librería ESPSoftware Serial de Dirk Kaar para permitir la comunicación serie del ESP32 y la librería de SparkFun (SparkFun_U_blox_GNSS_Arduino_Library.h[7]) de la que se ha obtenido el ejemplo del que se ha partido para desarrollar la programación del ESP32.

[6] En el caso del ESP32-C3-DevKit-02 la placa es la DOIT ESP32 DEVKIT V1.
[7] Enlace de descarga en el anexo.

Finalmente, es necesario modificar el *secrets.h* para introducir el nombre y la contraseña del *mountpoint* y del wifi. Además, se deben introducir las coordenadas geocéntricas de la base en la orden *myGNSS.setStaticPosition* que puede verse en la Figura 17.

```
80      myGNSS.setStaticPosition(468188940, 10, -13771846, -80, 431535712, 70);
81
82  Serial.println(F("Static position set"));
```

Figura 17. Instrucción en la que se establece la posición de la base.

4.3 Establecimiento de un caster propio

A pesar de no ser indispensable, si resulta recomendable la creación de un caster NTRIP propio en caso de desear establecer la comunicación con un gran número de clientes. Al emplear servicios como el que ofrece de manera gratuita SNIP con RTK2GO, se corre el riesgo de que la calidad de la señal se vea afectada por problemas de congestión de red y limitación de capacidad. Además, hay que tener en cuenta que el nivel de seguridad de los caster públicos suele ser más bajo, lo que hace más fácil la intercepción y manipulación de los datos de corrección.

Se ha servido del caster NTRIP que se ejecuta bajo Docker desarrollado a partir del código abierto de la BKG, fundadora del NTRIP [11]. Docker es una herramienta de software de código abierto basado en contenedores que permite implementar aplicaciones sea cual sea el entorno [12].

En la Figura 18, se muestran los tres tipos de archivos sobre los que está compuesto la estructura de Docker: los *Docker files*, las imágenes y los ya citados contenedores. *Un Docker file* es un archivo de texto con un conjunto de instrucciones que se ejecutan sucesivamente para crear una imagen. La imagen es un paquete ejecutable que incluye todo lo necesario para ejecutar un contenedor, por lo que se puede definir como la plantilla del contenedor. Finalmente, el contenedor es una unidad compacta y portátil que permite iniciar una aplicación.

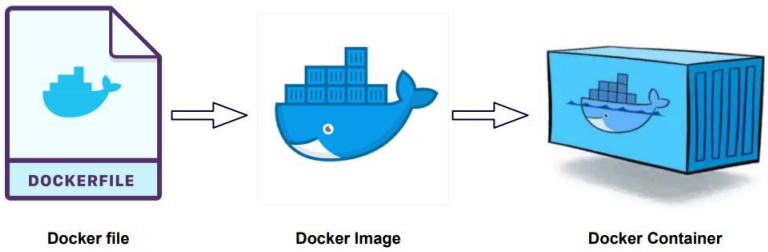


Figura 18. Archivos que componen la estructura de Docker. [13]

4.3.1 Ejecución del Docker

El primer paso para poder ejecutar un Docker es el descargar una aplicación que permita la implementación de aplicaciones Docker en un entorno local. Para este propósito, se va a emplear Docker Desktop[8].

Tras instalar y ejecutar la aplicación, se puede adquirir la imagen desde el terminal mediante el comando *docker pull*. Una vez escrito el comando es posible comprobar la imagen tanto desde Docker Desktop como con ayuda del comando *docker images*.

[8] Enlace de descarga en el anexo.

En la carpeta *testVolumen,* se incluyen los archivos que determinan las características del NTRIP caster que se va a ejecutar. Se debe modificar cada uno de estos archivos para controlar las siguientes características:

- ***Ntripcaster.conf***: Este archivo detalla características fundamentales como la dirección en la que se va a ubicar el caster, el puerto por el que se comunica o el número de clientes y base máximos que se pueden conectar al caster.
- ***Sourcetable.dat***: Se da la información relevante de cada uno de los *mountpoints* (nombre, tipo de mensaje a transmitir, ubicación, satélites de los que recibe información, …).
- ***Users.aut***: En el que se enumeran los usuarios con sus respectivas contraseñas.
- ***Groups.aut***: Los usuarios se asocian en los distintos grupos en los que se dividen en usuarios estándar, uploaders y administradores. También existe la posibilidad de crear nuevos grupos según las necesidades particulares.
- ***Clientmounts.aut***: Este archivo especifica los grupos que tienen acceso como cliente a cada uno de los *mountpoints*.
- ***Sourcemounts.aut***: En el que se detalla los grupos que tienen acceso como base a cada uno de los *mountpoints*.

Para comenzar a ejecutar el contenedor se emplea el comando *docker run* que puede verse en la Figura 19. En la instrucción, se crea un contenedor con el nombre *ntripcaster* que se comunica a través del puerto 2101, estándar en la comunicación NTRIP. Además, el comando hace que el contenedor se reinicie en caso de que se detenga e incluye los archivos de configuración anteriormente descritos como un volumen dentro del contenedor. Finalmente, con ayuda del comando *docker ps* es posible comprobar que el contenedor *ntripcaster* se esté ejecutando.

```
PS C:\> docker run -d -v C:\Users\iartajo\testVolumen:/etc/ntripcaster/
 -p 2101:2101 --name ntripcaster --restart=always rinex20/ntripcaster
```

Figura 19. Instrucción para ejecutar el caster.

4.3.2 Amazon Web Services (AWS)

En lugar de subir el caster a un sitio web en un servidor tradicional, se ha escogido ejecutarlo en *Amazon Web Services* (AWS). AWS es una plataforma de computación en la nube que ofrece acceso bajo demanda a una amplia gama de servicios compartidos a través de Internet. Se ha optado por alojar el caster en un servidor en la nube principalmente por la seguridad que ofrece, ya que entre las diversas medidas de seguridad utilizadas se encuentran el cifrado de datos o el empleo de técnicas de autenticación y autorización. Además, tiene la ventaja de poder escalar fácilmente la capacidad de cómputo y de almacenamiento del servidor.

Para poder ejecutar el caster, se requieren de 3 servicios de AWS: *Elastic Compute Cloud* (EC2*), Elastic Container Registry* (ECR) y *Elastic Container Service* (ECS). EC2 es un servicio de cómputo en la nube en la que se lanzan servidores virtuales conocidos en AWS como instancias. En este caso, se va a utilizar una instancia para ejecutar en su interior el Docker que contiene la aplicación del caster. ECR es un servicio que proporciona registros de almacenamiento denominados repositorios en los que se puede almacenar la imagen del Docker a utilizar. Finalmente, ECS es un servicio de orquestación de contenedores en el que la unidad fundamental de trabajo son las tareas. Una o más tareas pueden agruparse en un clúster, un grupo lógico de recursos que se utiliza para ejecutar y administrar contenedores. En la Figura 20, se muestra un esquema de los servicios de AWS utilizados y sus interacciones.

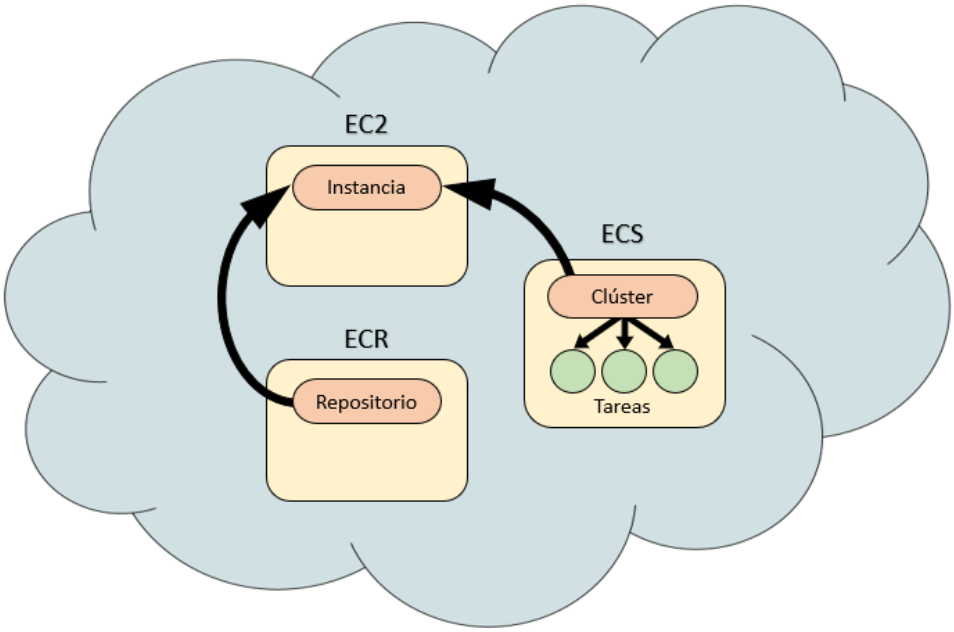


Figura 20. Esquema de servicios de AWS.

Los servicios de AWS se comunican entre sí según las reglas establecidas en los grupos de seguridad. Además, hay que tener en cuenta que para que se puedan comunicar entre los distintos servicios es necesario que se encuentren en la misma región[9].

Por tanto, lo primero que se debe hacer es crear el grupo de seguridad en EC2 que gestione la comunicación entre los distintos servicios de AWS. El grupo de seguridad se configura para que las reglas de entradas habiliten la transmisión de los puertos con los que se va a comunicar el caster, como puede verse en la Figura 21, mientras que las reglas de salida permitan todo el tráfico.

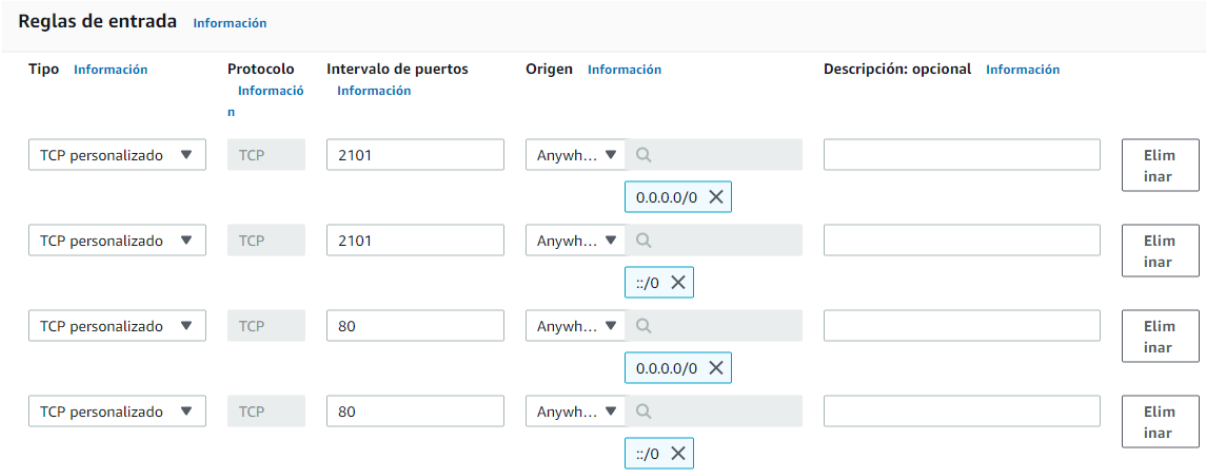


Figura 21. Reglas de entrada del grupo de seguridad.

A continuación, se debe lanzar la instancia desde EC2, en el que se debe escoger la capacidad de almacenamiento, el tipo de instancia t2.micro y el sistema operativo[10]. Además, hay que seleccionar las claves para el inicio de sesión por SSH con el que poder entrar a la instancia con el comando que se puede ver en la Figura 22. Una vez dentro de la instancia, se instala Docker en el sistema operativo de la instancia y se introducen los archivos de configuración necesarios para poder ejecutar el caster.

```
ssh -i "claves" user@IP
```

Figura 22. Comando para iniciar sesión en la instancia creada en AWS.

[9] Se ha empleado la región de Europa Central.
[10] Se ha utilizado un Linux, aunque es posible utilizar un sistema Windows.

Seguidamente, se debe crear un repositorio público en ECR que se va a utilizar para almacenar la imagen del Docker, como el que se puede ver en la Figura 23. Tras crear el repositorio, es necesario descargar la última versión de AWS CLI[11] y una vez instalado desde el terminal se debe configurar la cuenta de AWS con ayuda del comando *aws configure*. Dentro del repositorio, en *Ver comandos de envío* se pueden ver los pasos a seguir para subir la imagen al repositorio, no obstante, en lugar de realizar un *docker build* se debe realizar el *docker pull* y seguidamente un *docker tag* para cambiarle el nombre a la imagen por el nombre del repositorio.

Figura 23. Repositorio público para almacenar la imagen del Docker.

Una vez generado el repositorio, se debe definir la tarea que ejecute la imagen que se encuentra en el repositorio. En la sección de Definición de tareas de ECS, se crea una nueva definición de tarea en un JSON. Es importante definirlo de manera precisa, incluyendo la imagen del repositorio, los puertos desde los que se van a realizar la comunicación por protocolo TCP, la memoria utilizada o los archivos de configuración del caster incluidos dentro de un volumen. Se puede encontrar el JSON que define la tarea a realizar en el apartado de anexos. Finalmente, dentro de la sección de tareas del Clúster hay que seleccionar *Ejecutar nueva tarea*.

Por último, se crea el Clúster en ECS donde se elige el mismo tipo de instancia y sistema operativo que al crear la instancia. Es importante también tener en cuenta la subred en la que se van a ejecutar las tareas, escoger el par de claves utilizados para conectarse a la instancia por SSH y asignar el grupo de seguridad anteriormente creado.

[11] Enlace de descarga en el anexo.

4.4 Transmisión por radio

Este apartado se enfoca en la descripción de la transmisión del flujo de corrección desde una estación base a distintas estaciones móviles a través de ondas de radio. A diferencia del protocolo NTRIP, la transmisión por radio no requiere de una red celular ni de un caster que actúe como nexo en la comunicación entre las bases y los vehículos autónomos.

4.4.1 Componentes

En este caso, la base está compuesta por el módulo de comunicación por radiofrecuencia de largo alcance XBee SX868, mostrado en la Figura 24, además de por la placa de desarrollo SimpleRTK2B y la antena ANN-MB-00-00 anteriormente vistas. El XBee SX868 opera en la banda de frecuencia de 868 MHz, de uso público dentro de la Unión Europea y principalmente utilizada para dispositivos de uso industrial, científico y médico (ISM) [14].

Figura 24. XBee SX868.

Además de un dispositivo con el que poder configurar la base a través de U-center, también es necesario definir las características del módulo de comunicación mediante XCTU. XCTU es una herramienta de software desarrollada por Digi International para la configuración y el monitoreo de módulos XBee. Para poder conectar el XBee al ordenador es necesario un adaptador como el que se puede ver en la Figura 25.

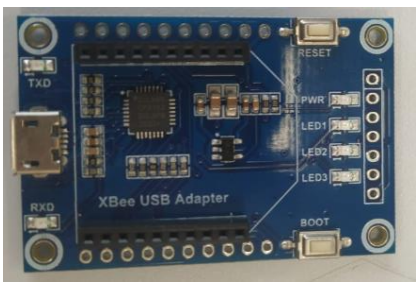


Figura 25. Adaptador para la conexión del XBee por USB.

4.4.2 Configuración

Tras establecer las coordenadas de la base en U-center, se seleccionan los mensajes que se van a enviar. En este caso, se van a emitir los mismos mensajes RTCM que en el apartado 4.2.1.1, ya que estos son más compactos que los mensajes NMEA y pueden ser transmitidos más eficientemente a través de canales de comunicación con ancho de banda limitado. Sin embargo, se va a definir la comunicación en lugar de por USB a través de UART2, por el que se va a conectar el XBee.

A continuación, se debe instalar XCTU[12] y conectar el XBee al ordenador con ayuda del adaptador. Seguidamente se selecciona el icono de añadir dispositivo que aparece arriba a la izquierda de la aplicación para añadir el módulo a la lista. Para poder encontrar el dispositivo, es necesario designar los parámetros con los que trabaja. Tras unos segundos de espera, se debería encontrar el módulo, en caso contrario se debe comprobar si los parámetros introducidos son con los que opera el módulo.

Una vez que el dispositivo este añadido en la lista, al seleccionarlo se accede a los ajustes de red en el que se configura su transmisión. Para poder establecer la comunicación es imprescindible que el canal (CH), el ID y el *baud rate* (BD) sean iguales en todos los dispositivos que se quieran comunicar [15]. En el caso de la base, como se desea que se comunique a todos los dispositivos conectados a la red se debe dejar la *Destination Address High* (DH) en 0 y la *Destination Address Low* (DL) igual a FFFF. De igual forma, es importante establecer a la base como coordinador (CE) y a los clientes como dispositivos finales. La base se comunica a través de UART con el XBee, por lo que se debe verificar que el pin DIN esté como entrada y el pin DOUT como salida. Finalmente, se debe guardar los cambios realizados presionando el icono *Write*.

Para verificar que la comunicación entre ambos módulos sea posible, se puede acceder al modo de funcionamiento de las consolas desde el icono de arriba a la derecha de la aplicación. Al abrir la comunicación de los dos puertos de comunicación es posible intercambiar mensajes desde la consola.

[12] Enlace de descarga en el anexo.

5 Resultados y discusión

En el siguiente apartado, se van a presentar los resultados obtenidos en cada uno de los puntos establecidos en el apartado anterior, para seguidamente, discutir los valores logrados.

5.1 Posicionamiento de la base

Es posible comprobar la precisión de la posición de la base en los resultados proporcionados por el centro de procesamiento CSRS. En el caso presentado a continuación, se midió la posición en un entorno cerrado junto a una ventana lo que resulta completamente desaconsejable, ya que se necesita un entorno abierto sin obstáculos cercanos para poder medir la posición de manera precisa. Se estuvo calculando la posición durante casi 11 horas (10:53:28) y como se puede apreciar en la Figura 26 a pesar de poder concretar la posición de la base con las coordenadas obtenidas, no es lo suficiente precisa como para que corrija la posición de un vehículo para que circule entre dos hileras de cultivos.

The estimated coordinates **ITRF20 2023-02-08** for the **COM7___9600_230208_100500.obs** RINEX file are as follows:

Latitude N42° 50' 47.8761" ± 1.445 m (95%)
Longitude W1° 41' 05.4958" ± 2.418 m (95%)
Ellipsoidal Height 490.300 m ± 2.997 m (95%)
[42.84663225,-1.68485995,490.300]

UTM Zone 30 (North)
Northing 4744622.575 m
Easting 607461.848 m
Scale factor (point) 0.99974207
Scale factor (combined) 0.99966520
[4744622.575,607461.848,490.300]

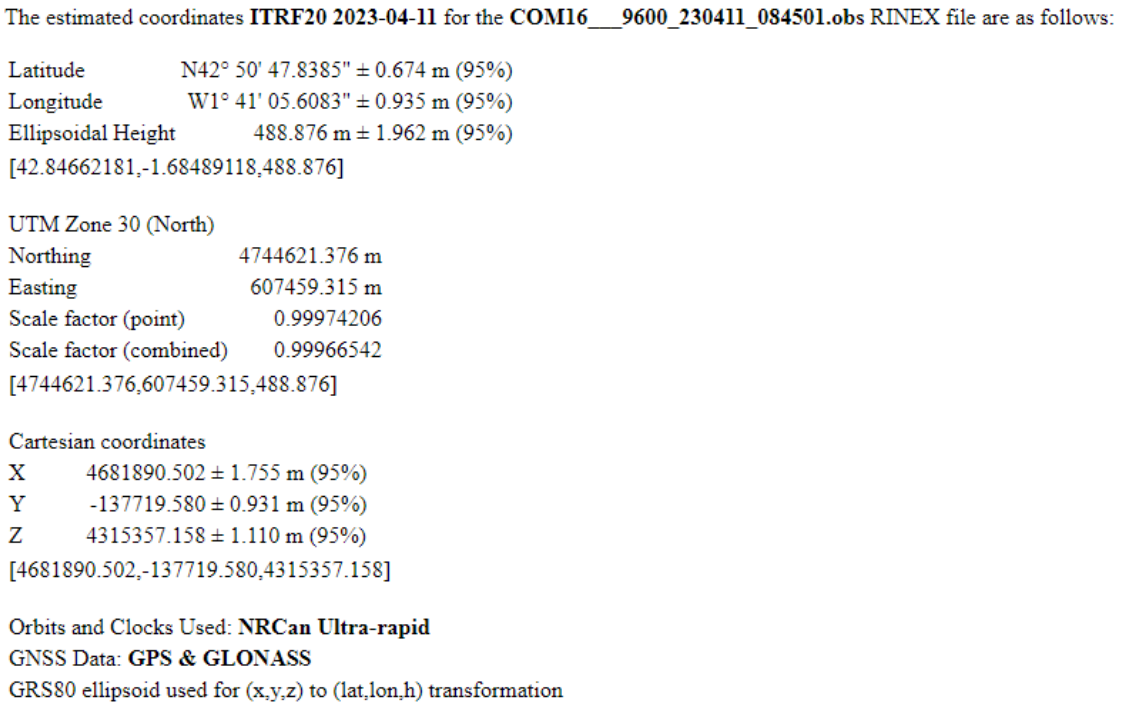
Cartesian coordinates
X 4681890.833 ± 2.663 m (95%)
Y -137717.036 ± 2.395 m (95%)
Z 4315358.977 ± 2.024 m (95%)
[4681890.833,-137717.036,4315358.977]

Orbits and Clocks Used: **NRCan Rapid**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 26. Medición de 11 horas con la antena ANN-MB-00-00.

De los resultados obtenidos también se puede observar que la posición horizontal tiene una mayor precisión que la posición vertical, debido a que las constelaciones de satélites están diseñadas en mayor medida para proporcionar una posición horizontal y que la atmosfera terrestre es menos homogénea en la dirección vertical que en la horizontal, lo que ocasiona que los efectos atmosféricos originen un error vertical mayor.

Como se observó en la Figura 8, al aumentar el tiempo de medición el error en la posición se reduce, ya que cuanto mayor tiempo tenga la medición hay un mayor número de muestras con el que poder realizar el promedio. En este caso, al pasar de las casi 11 horas a las 15 horas (15:03:55) de la medición de la Figura 27 se puede ver como los errores de la medición se han reducido, aunque el error vertical siga siendo mayor que el horizontal. Sin embargo, sigue existiendo un error considerable al medir en un entorno cerrado.



The estimated coordinates **ITRF20 2023-04-11** for the **COM16___9600_230411_084501.obs** RINEX file are as follows:

Latitude N42° 50' 47.8385" ± 0.674 m (95%)
Longitude W1° 41' 05.6083" ± 0.935 m (95%)
Ellipsoidal Height 488.876 m ± 1.962 m (95%)
[42.84662181,-1.68489118,488.876]

UTM Zone 30 (North)
Northing 4744621.376 m
Easting 607459.315 m
Scale factor (point) 0.99974206
Scale factor (combined) 0.99966542
[4744621.376,607459.315,488.876]

Cartesian coordinates
X 4681890.502 ± 1.755 m (95%)
Y -137719.580 ± 0.931 m (95%)
Z 4315357.158 ± 1.110 m (95%)
[4681890.502,-137719.580,4315357.158]

Orbits and Clocks Used: **NRCan Ultra-rapid**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 27. Medición de 15 horas con la antena ANN-MB-00-00.

En el apartado de anexos, se ha adjuntado de manera más detallada los resultados de esta última medición en las que se puede ver que la base detecta a los satélites con los que trabajaba de manera discontinua, a causa de los obstáculos, al estar ubicado en un entorno cerrado.

También, se ha calculado la posición empleando la antena AS-ANT2B-SUR-L1L2-25SMA-00 de mayores prestaciones que la antena ANN-MB-00-00. En la Figura 28, resulta evidente que el error obtenido con esta antena es considerablemente inferior al logrado con la ANN-MB-00-00. Por si fuera poca la diferencia del error de la posición entre estas dos antenas, cabe destacar que la medición de esta antena se ha realizado en menos de 7 horas (6:39:01).

The estimated coordinates **ITRF20 2023-02-16** for the **Antenabuena.obs** RINEX file are as follows:

Latitude N42° 50' 47.8649" ± 0.120 m (95%)
Longitude W1° 41' 05.5540" ± 0.281 m (95%)
Ellipsoidal Height 488.249 m ± 0.292 m (95%)
[42.84662913,-1.68487610,488.249]

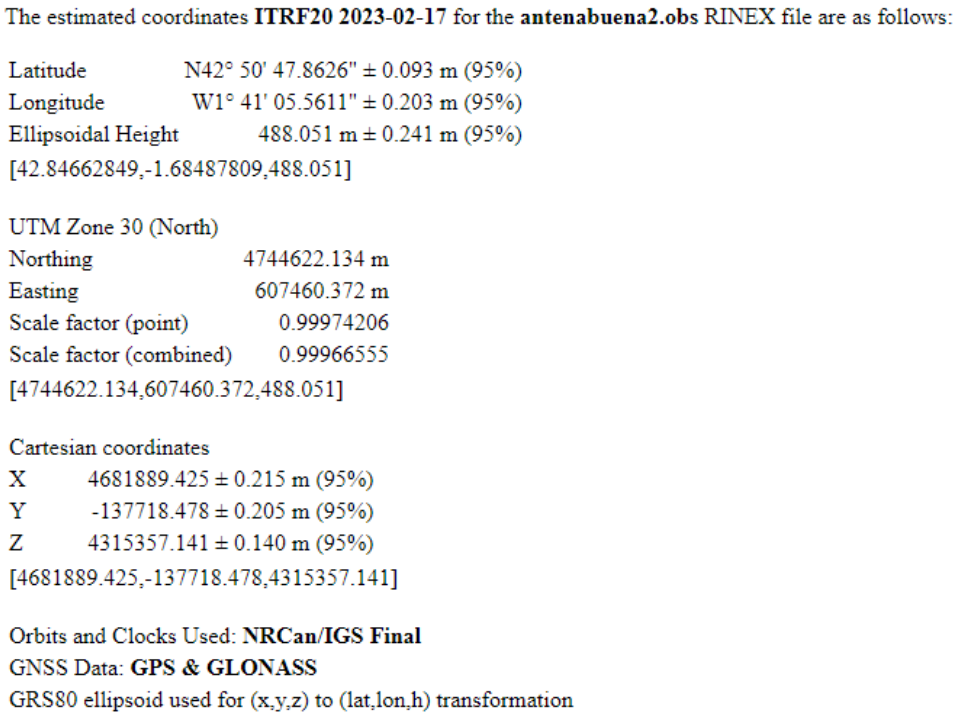
UTM Zone 30 (North)
Northing 4744622.207 m
Easting 607460.534 m
Scale factor (point) 0.99974207
Scale factor (combined) 0.99966552
[4744622.207,607460.534,488.249]

Cartesian coordinates
X 4681889.527 ± 0.270 m (95%)
Y -137718.318 ± 0.282 m (95%)
Z 4315357.327 ± 0.161 m (95%)
[4681889.527,-137718.318,4315357.327]

Orbits and Clocks Used: **NRCan Ultra-rapid**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 28. Medición de 7 horas con la antena AS-ANT2B-SUR-L1L2-25SMA-00.

En la Figura 29, se puede comprobar que al igual que con la otra antena al aumentar el tiempo de medición el error en la posición se reduce. En este caso, se ha pasado de las casi 7 horas de la medición anterior a las 13 horas (13:17:13) de la medición de la siguiente imagen.



The estimated coordinates **ITRF20 2023-02-17** for the **antenabuena2.obs** RINEX file are as follows:

Latitude N42° 50' 47.8626" ± 0.093 m (95%)
Longitude W1° 41' 05.5611" ± 0.203 m (95%)
Ellipsoidal Height 488.051 m ± 0.241 m (95%)
[42.84662849,-1.68487809,488.051]

UTM Zone 30 (North)
Northing 4744622.134 m
Easting 607460.372 m
Scale factor (point) 0.99974206
Scale factor (combined) 0.99966555
[4744622.134,607460.372,488.051]

Cartesian coordinates
X 4681889.425 ± 0.215 m (95%)
Y -137718.478 ± 0.205 m (95%)
Z 4315357.141 ± 0.140 m (95%)
[4681889.425,-137718.478,4315357.141]

Orbits and Clocks Used: **NRCan/IGS Final**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 29. Medición de 13 horas con la antena AS-ANT2B-SUR-L1L2-25SMA-00.

En la Figura 30, se muestra una medición de aproximadamente 24 horas (23:32:22) realizada desde un tejado. A diferencia de las mediciones anteriores, esta se llevó a cabo en un espacio sin obstáculos que interfiriesen con la señal satelital, lo que resulto en una reducción del error de posición al orden del milímetro. En esta medición, se puede constatar una vez más como el error vertical es superior al error horizontal al medir la ubicación de la base

The estimated coordinates **ITRF20 2023-05-18** for the **COM4___9600_230518_161048.obs** RINEX file are as follows:

Latitude N42° 45' 37.1326" ± 0.003 m (95%)
Longitude W1° 34' 02.4901" ± 0.003 m (95%)
Ellipsoidal Height 571.368 m ± 0.012 m (95%)
[42.76031462,-1.56735835,571.368]

UTM Zone 30 (North)
Northing 4735193.840 m
Easting 617226.098 m
Scale factor (point) 0.99976906
Scale factor (combined) 0.99967949
[4735193.840,617226.098,571.368]

Cartesian coordinates
X 4688736.520 ± 0.009 m (95%)
Y -128295.035 ± 0.003 m (95%)
Z 4308378.185 ± 0.009 m (95%)
[4688736.520,-128295.035,4308378.185]

Orbits and Clocks Used: **NRCan Ultra-rapid**
GNSS Data: **GPS & GLONASS**
GRS80 ellipsoid used for (x,y,z) to (lat,lon,h) transformation

Figura 30. Medición de 24 horas en un entorno abierto.

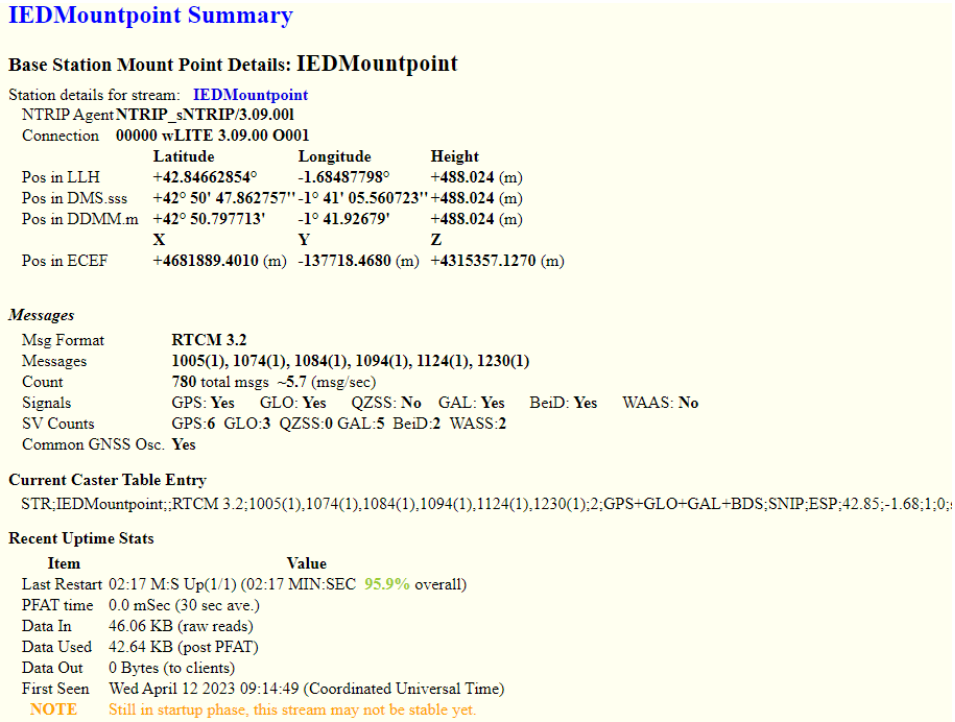
Tras analizar los datos obtenidos se puede ver la importancia que tiene trabajar en un entorno abierto sin obstáculos que puedan interferir en las señales satelitales. Al trabajar dentro de un edificio, aunque sea pegado a una ventana, el error de posición aumenta considerablemente no pudiendo utilizar la base para corregir la posición de un vehículo que circule a través de hileras de cultivos. Además, se ha comprobado que el error vertical es superior al error horizontal, que la antena AS-ANT2B-SUR-L1L2-25SMA-00 tiene una mayor precisión que la antena ANN-MB-00-00 y que cuanto mayor es el tiempo de medición menor es el error obtenido.

5.2 Transmisión por Internet

Para poder comprobar la correcta transmisión de la base se va a emplear Mission Planner[13], simulando a un posible cliente que reciba el flujo de corrección proporcionado por la base. Mission Planner es un software de código abierto utilizado para controlar misiones de vehículos aéreos no tripulados.

5.2.1 Transmisión con ordenador

Al comunicar la base con el caster, el nombre del *mountpoint* al que se atribuya la base comienza a aparecer en la página web del caster. En el caso de RTK2GO, se puede acceder a un resumen de la comunicación como el que se ve en la Figura 31 en la que aparecen entre otros valores las coordenadas de la base, el formato del mensaje y los satélites con los que se comunica la base.



IEDMountpoint Summary

Base Station Mount Point Details: IEDMountpoint

Station details for stream: **IEDMountpoint**

NTRIP Agent	**NTRIP_sNTRIP/3.09.001**		
Connection	**00000 wLITE 3.09.00 O001**		
	Latitude	**Longitude**	**Height**
Pos in LLH	**+42.84662854°**	**-1.68487798°**	**+488.024** (m)
Pos in DMS.sss	**+42° 50' 47.862757"**	**-1° 41' 05.560723"**	**+488.024** (m)
Pos in DDMM.m	**+42° 50.797713'**	**-1° 41.92679'**	**+488.024** (m)
	X	**Y**	**Z**
Pos in ECEF	**+4681889.4010** (m)	**-137718.4680** (m)	**+4315357.1270** (m)

Messages

Msg Format	**RTCM 3.2**
Messages	**1005(1), 1074(1), 1084(1), 1094(1), 1124(1), 1230(1)**
Count	**780** total msgs ~**5.7** (msg/sec)
Signals	GPS: **Yes** GLO: **Yes** QZSS: **No** GAL: **Yes** BeiD: **Yes** WAAS: **No**
SV Counts	GPS:**6** GLO:**3** QZSS:**0** GAL:**5** BeiD:**2** WASS:**2**
Common GNSS Osc.	**Yes**

Current Caster Table Entry

STR;IEDMountpoint;;RTCM 3.2;1005(1),1074(1),1084(1),1094(1),1124(1),1230(1);2;GPS+GLO+GAL+BDS;SNIP;ESP;42.85;-1.68;1;0;

Recent Uptime Stats

Item	Value
Last Restart	02:17 M:S Up(1/1) (02:17 MIN:SEC 95.9% overall)
PFAT time	0.0 mSec (30 sec ave.)
Data In	46.06 KB (raw reads)
Data Used	42.64 KB (post PFAT)
Data Out	0 Bytes (to clients)
First Seen	Wed April 12 2023 09:14:49 (Coordinated Universal Time)
NOTE	Still in startup phase, this stream may not be stable yet.

Figura 31. Resumen de la comunicación de la base con el caster en RTK2GO.

[13] Enlace de descarga en el anexo.

Para asegurarse que la comunicación resulta lo suficientemente fiable como para que corrija la posición de un vehículo, es posible comprobar la comunicación con un cliente simulado través de Mission Planner. Para ello, se debe introducir la url con la información necesaria para que el cliente se conecte al caster (usuario, contraseña, nombre del *mountpoint*, puerto y host) en *SETUP -> Optional Hardware -> RTK/GPS Inject*. En la Figura 32, se puede ver como la comunicación es óptima para recibir el flujo de corrección, mostrando los mensajes que le van llegando al posible cliente en cada momento.

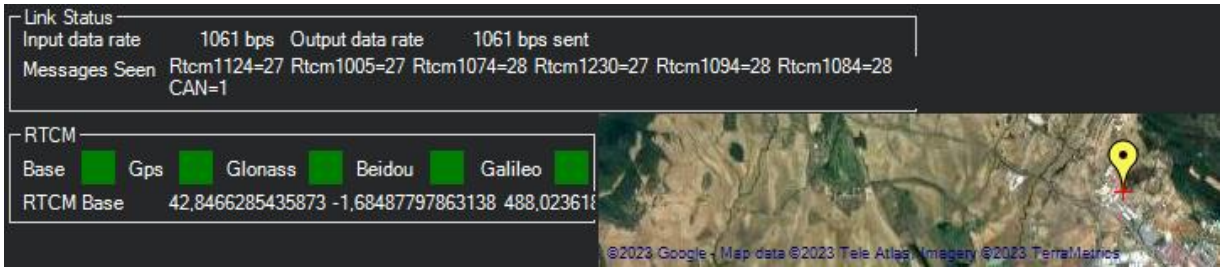
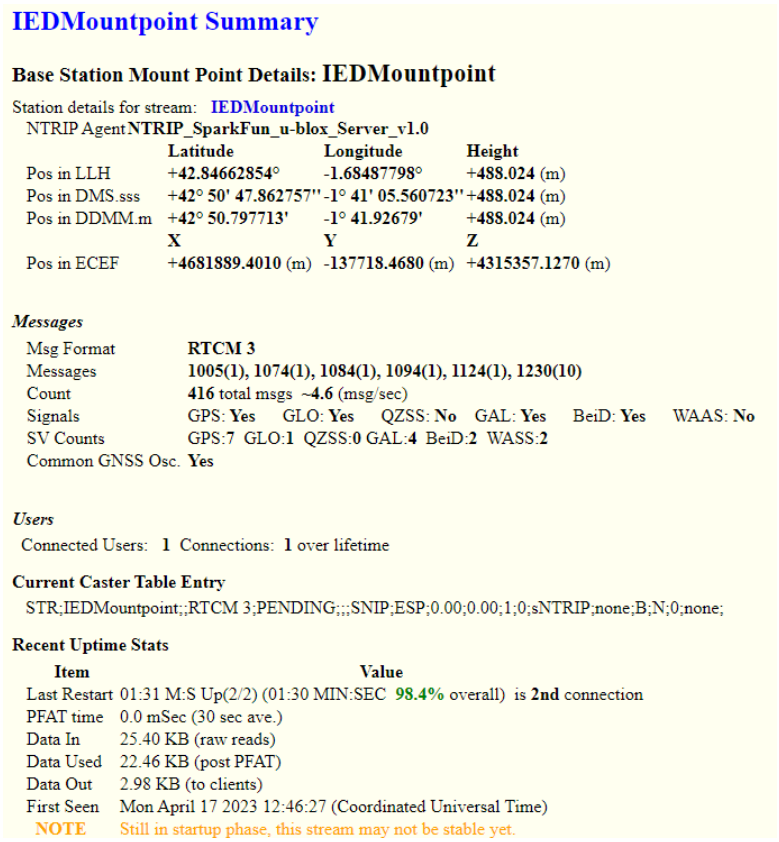


Figura 32. Simulación de cliente de la transmisión por ordenador.

5.2.2 Transmisión por wifi

En un comienzo se trabajó con una comunicación por I2C entre el SimpleRTK2B y el ESP32. Sin embargo, a pesar de que como puede verse en la Figura 33 la base lograba comunicarse correctamente con el caster, esta no era capaz de recuperar la transmisión en el momento en el que perdía la alimentación. Por lo que se recomienda, conectar la base además de a la red eléctrica a una fuente de alimentación externa que asegure la comunicación ante pérdidas de energía.



IEDMountpoint Summary

Base Station Mount Point Details: IEDMountpoint

Station details for stream: IEDMountpoint

NTRIP Agent **NTRIP_SparkFun_u-blox_Server_v1.0**

	Latitude	Longitude	Height
Pos in LLH	**+42.84662854°**	**-1.68487798°**	**+488.024** (m)
Pos in DMS.sss	**+42° 50' 47.862757''**	**-1° 41' 05.560723''**	**+488.024** (m)
Pos in DDMM.m	**+42° 50.797713'**	**-1° 41.92679'**	**+488.024** (m)
	X	**Y**	**Z**
Pos in ECEF	**+4681889.4010** (m)	**-137718.4680** (m)	**+4315357.1270** (m)

Messages

Msg Format	**RTCM 3**
Messages	**1005(1), 1074(1), 1084(1), 1094(1), 1124(1), 1230(10)**
Count	**416** total msgs ~**4.6** (msg/sec)
Signals	GPS: **Yes** GLO: **Yes** QZSS: **No** GAL: **Yes** BeiD: **Yes** WAAS: **No**
SV Counts	GPS:**7** GLO:**1** QZSS:**0** GAL:**4** BeiD:**2** WASS:**2**
Common GNSS Osc.	**Yes**

Users

Connected Users: **1** Connections: **1** over lifetime

Current Caster Table Entry

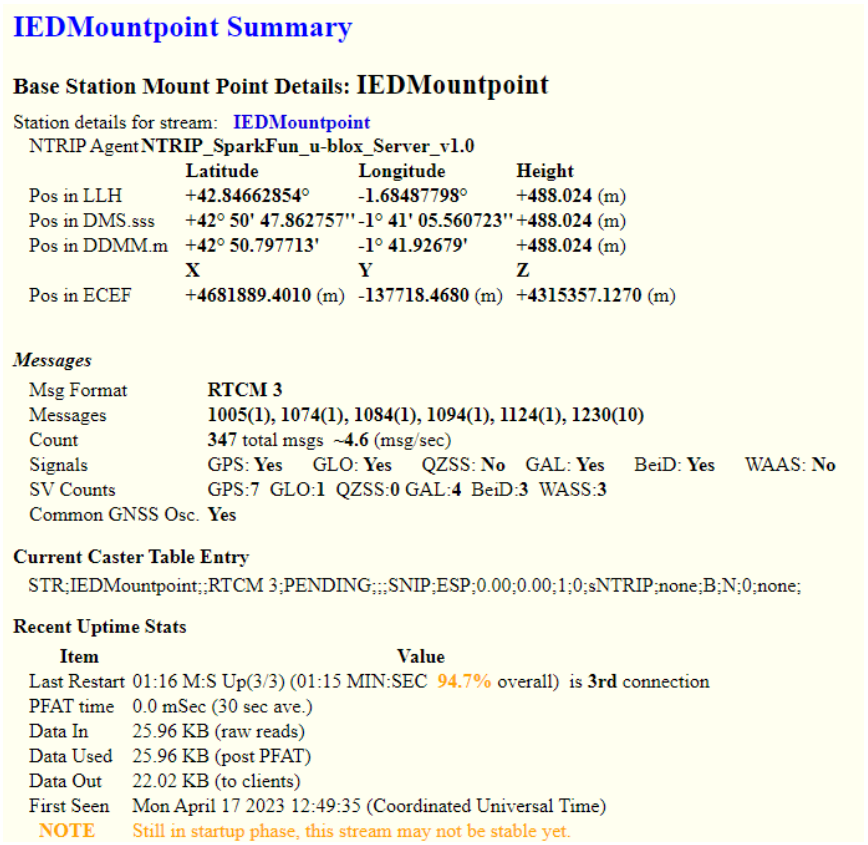
STR;IEDMountpoint;;RTCM 3;PENDING;;;SNIP;ESP;0.00;0.00;1;0;sNTRIP;none;B;N;0;none;

Recent Uptime Stats

Item	Value
Last Restart	01:31 M:S Up(2/2) (01:30 MIN:SEC **98.4%** overall) is **2nd** connection
PFAT time	0.0 mSec (30 sec ave.)
Data In	25.40 KB (raw reads)
Data Used	22.46 KB (post PFAT)
Data Out	2.98 KB (to clients)
First Seen	Mon April 17 2023 12:46:27 (Coordinated Universal Time)
NOTE	Still in startup phase, this stream may not be stable yet.

Figura 33. Resumen de la comunicación con el caster en RTK2GO con I2C.

Ante la falta de fiabilidad del sistema, posiblemente causada por el tiempo de inicio que requiere este protocolo, se optó por mantener una comunicación Serial. Este protocolo logra mantener una comunicación eficaz con el caster emitiendo los mismos mensajes que con el protocolo I2C, no obstante, es capaz de recuperar la comunicación con el caster ante posibles apagones de alimentación. En la Figura 34, se muestra el resumen de la comunicación en RTK2GO con este protocolo.



IEDMountpoint Summary

Base Station Mount Point Details: IEDMountpoint

Station details for stream: **IEDMountpoint**

NTRIP Agent **NTRIP_SparkFun_u-blox_Server_v1.0**

	Latitude	Longitude	Height
Pos in LLH	**+42.84662854°**	**-1.68487798°**	**+488.024** (m)
Pos in DMS.sss	**+42° 50' 47.862757"**	**-1° 41' 05.560723"**	**+488.024** (m)
Pos in DDMM.m	**+42° 50.797713'**	**-1° 41.92679'**	**+488.024** (m)
	X	**Y**	**Z**
Pos in ECEF	**+4681889.4010** (m)	**-137718.4680** (m)	**+4315357.1270** (m)

Messages

Msg Format	**RTCM 3**
Messages	**1005(1), 1074(1), 1084(1), 1094(1), 1124(1), 1230(10)**
Count	**347** total msgs ~**4.6** (msg/sec)
Signals	GPS: **Yes** GLO: **Yes** QZSS: **No** GAL: **Yes** BeiD: **Yes** WAAS: **No**
SV Counts	GPS:**7** GLO:**1** QZSS:**0** GAL:**4** BeiD:**3** WASS:**3**
Common GNSS Osc.	**Yes**

Current Caster Table Entry

STR;IEDMountpoint;;RTCM 3;PENDING;;;SNIP;ESP;0.00;0.00;1;0;sNTRIP;none;B;N;0;none;

Recent Uptime Stats

Item	Value
Last Restart	01:16 M:S Up(3/3) (01:15 MIN:SEC **94.7%** overall) is **3rd** connection
PFAT time	0.0 mSec (30 sec ave.)
Data In	25.96 KB (raw reads)
Data Used	25.96 KB (post PFAT)
Data Out	22.02 KB (to clients)
First Seen	Mon April 17 2023 12:49:35 (Coordinated Universal Time)
NOTE	Still in startup phase, this stream may not be stable yet.

Figura 34. Resumen de la comunicación con el caster en RTK2GO con Serial.

En la Figura 35, se puede ver la simulación en Mission Planner de un cliente que recibe el flujo de corrección de la base con protocolo Serial. Como se puede apreciar, la comunicación establecida tiene una menor transferencia de datos que al utilizar un ordenador al no transmitir los mensajes NMEA. A diferencia de al emplear SNIP, la base no requiere de un ordenador para operar, aunque todavía requiera de una fuente de alimentación y una conexión wifi para su desempeño.

Figura 35. Simulación de un cliente con una Base con protocolo Serial.

5.3 Establecimiento de un caster propio

En un principio, al encontrar el código abierto original en Docker de la BKG se planteó la idea de trabajar con una máquina virtual con Linux, ya que únicamente se había comprobado su funcionamiento en este tipo de sistemas. Una máquina virtual es un contenedor de software aislado en el cual se puede ejecutar, de igual forma, tanto sistemas operativos como aplicaciones, siendo similar a AWS, pero sin la conectividad que ofrece estar en la nube. Sin embargo, debido a lo pesado que resulta ejecutar un nuevo sistema operativo aislado en un ordenador, se decidió trabajar con el Docker *rinex20/ntripcaster* encontrado en Docker Hub que además de poder ejecutarlo en Windows contiene configuraciones más avanzadas como la posibilidad de dividir a los clientes en distintos grupos o poder contar con administradores.

Antes de ejecutar el Docker en AWS, se verificó el funcionamiento del caster mediante la dirección IP local del dispositivo (127.0.0.1). De esta forma, al utilizar SNIP con la configuración de la Figura 36, se comprobó que la base se conecta al caster, siempre y cuando no se empleen usuarios, ya que la versión gratuita de SNIP no dispone del uso de usuarios.

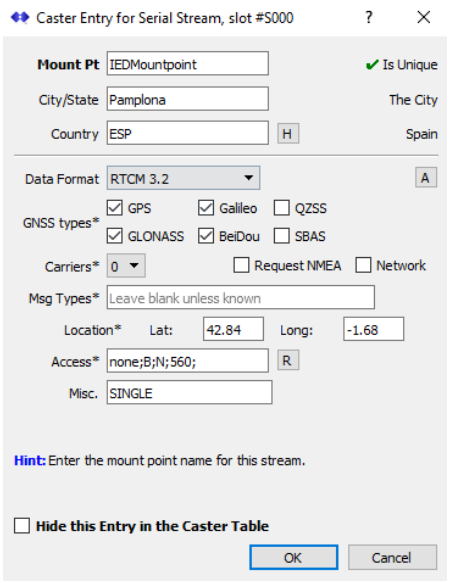


Figura 36. Configuración de SNIP para conectar con el caster que se ejecuta en Docker.

En el caso de la comunicación por wifi, hay que tener en cuenta que, si el ESP32 y el ordenador están conectados a redes distintas, es probable que no puedan comunicarse entre sí. Esto se debe a que cada dispositivo tiene asignada una dirección IP en su respectiva red que no suele ser accesible desde otras redes. Para establecer comunicación, es posible conectar ambos dispositivos a la misma red. Sin embargo, es importante mencionar que, si se planea instalar bases muy distantes entre sí, esta solución implicaría la necesidad de contar con más de un ordenador.

Para comprobar el correcto funcionamiento del caster, se empleó el comando *docker exec* desde el terminal, con el cual se accedió al contenedor en ejecución en el que se encontraban varios archivos de registro, entre los que se destacan *ntripcaster.log* y *access.log* mostrados en la Figura 37. En el archivo *ntripcaster.log*, se muestran las características principales del caster y de cada una de las conexiones que se realizan en ella. Mientras que en *access.log*, se muestran de manera periódica el ancho de banda y el número de clientes y bases conectadas.

```
cat ntripcaster-230509.log
] [0:Main Thread] Using stdout as NtripCaster logging window
] [0:Main Thread] Starting main connection handler...
] [0:Main Thread] Listening on port 2101...
] [0:Main Thread] Using 'localhost' as servername...
] [0:Main Thread] Server limits: 1000 clients, 1000 clients per source, 40 sources, 2 admins
] [0:Main Thread] WWW Admin interface accessible at http://localhost:2101/admin
] [0:Main Thread] Starting Calender Thread...
] [0:Main Thread] Starting relay connector thread...
] [6:Connection Handler] Accepted encoder on mountpoint /IEDMountpoint from 172.17.0.1. 1 sources connected
] [7:Connection Handler] Accepted http client 1 from [172.17.0.1] on mountpoint [/IEDMountpoint]. 1 clients connected
] [8:Connection Handler] Accepted http client 2 from [172.17.0.1] on mountpoint [/IEDMountpoint]. 2 clients connected
cat usage-230509.log
] [1:Calendar Thread] [09/May/2023:10:07:44] Bandwidth:0.000000KB/s Sources:0 Clients:0 Admins:0
] [1:Calendar Thread] [09/May/2023:10:09:44] Bandwidth:0.733333KB/s Sources:1 Clients:1 Admins:0
```

Figura 37. Archivos de registro del Docker ejecutado en localhost

Tras verificar desde la dirección IP local el desempeño del caster, se ejecutó el Docker a través de una instancia de AWS. De este modo, es posible conectar desde cualquier ubicación una base, tanto si se conecta desde el ordenador como si se emplea un ESP32. Sin embargo, a pesar de que el caster permita el empleo de usuarios, al no disponer de usuarios en la versión gratuita de SNIP y al no incluirlos en el código utilizado en la ESP32, no fue posible utilizarlos.

El caster subido a AWS permite un gran volumen de tráfico limitado a 40 bases y 1000 clientes. En la Figura 38, se puede ver como se han conectado al caster una base y una docena de clientes, simulados a través de Mission Planner.

```
[09/May/2023:10:51:43] Bandwidth:2.700000KB/s Sources:1 Clients:12 Admins:0
```

Figura 38. Ancho de banda y número de clientes y bases conectados al caster.

A pesar de haber logrado un caster operativo en AWS que solventa los problemas de congestión en la red de los casters públicos, se han tenido algunas dificultades en la ejecución, ya que AWS tiene una curva de aprendizaje pronunciada para los nuevos usuarios en la plataforma. No obstante, a pesar de la complejidad resulta recomendable su uso ya que ofrece una mayor seguridad que un servidor en la nube y se puede escalar la capacidad de cómputo y almacenamiento en función de las necesidades del caster.

5.4 Transmisión por radio

A pesar de que en una primera impresión pueda parecer que la comunicación por radio es más sencilla de configurar que por NTRIP, se han tenido que afrontar varios contratiempos por incompatibilidades entre los módulos XBee y los adaptadores USB utilizados. Estos pueden ser debidos al uso de diferentes versiones del estándar de comunicación que pueden solventarse mediante actualizaciones del firmware.

Desgraciadamente, no se ha podido comprobar la calidad de la señal de radio y la comunicación por NTRIP en la zona de cultivo. No obstante, conociendo las características de cada una de las comunicaciones se pueden obtener varias consideraciones:

- Al trabajar en los cultivos es posible que el módulo de radiofrecuencia se encuentre por debajo de los arbustos lo que puede ocasionar interferencias en la comunicación por ondas de radio.
- Es posible que en ciertos momentos la cobertura de la señal de Internet no sea la suficiente estable como para poder establecer la comunicación por NTRIP.
- Aunque resulte poco probable, por la zona en la que se encuentra ubicada, puede que haya interferencias electromagnéticas que afecten tanto a la calidad de la señal de radio como a la señal del NTRIP.

5.5 Precisión posición del vehículo

Al no contar todavía con el vehículo agrónomo se ha decidido comprobar la precisión del sistema de corrección en la posición del vehículo utilizando un Vehículo de Guiado Automático (AGV) como el de la Figura 39. Este AGV es capaz de circular de manera autónoma sobre un recorrido establecido con cinta magnética.

Figura 39. Vehículo de Guiado Automático.

El vehículo graba su posición cada 3 segundos, lo cual permite obtener una muestra significativa después de varias horas como para poder comprobar el error en la precisión. Se ha optado por un recorrido con forma de "D" como el que puede verse en la Figura 40, con el fin de comprobar la precisión tanto en rectas como en curvas. La presencia de los codos se debe a la limitación del AGV para moverse en ángulos rectos.

Figura 40. Cinta magnética que marca el recorrido que realiza el AGV.

Para poder dibujar la longitud y latitud de cada uno de los puntos en un plano es necesario realizar una proyección cartográfica. Para ello, es posible calcular la distancia entre dos puntos de una superficie esférica mediante la fórmula de Haversine[14] [16].

$$d = 2\,R\,\arcsin\left(\sqrt{sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lon_1)\cos(lon_2)\ sin^2\left(\frac{lat_2 - lat_1}{2}\right)}\right) \quad \text{(ec. 5)}$$

Calculando la distancia latitudinal y longitudinal respecto a un punto de referencia, como puede ser el centro del recorrido, es posible dibujar cada una de las posiciones en un plano. En la Figura 41, se muestra las posiciones del vehículo empleando la antena ANN-MB-00-00 y se puede comprobar a pesar de tener puntos que se desvían del recorrido ninguno de ellos llega a alejarse más de 10 centímetros.

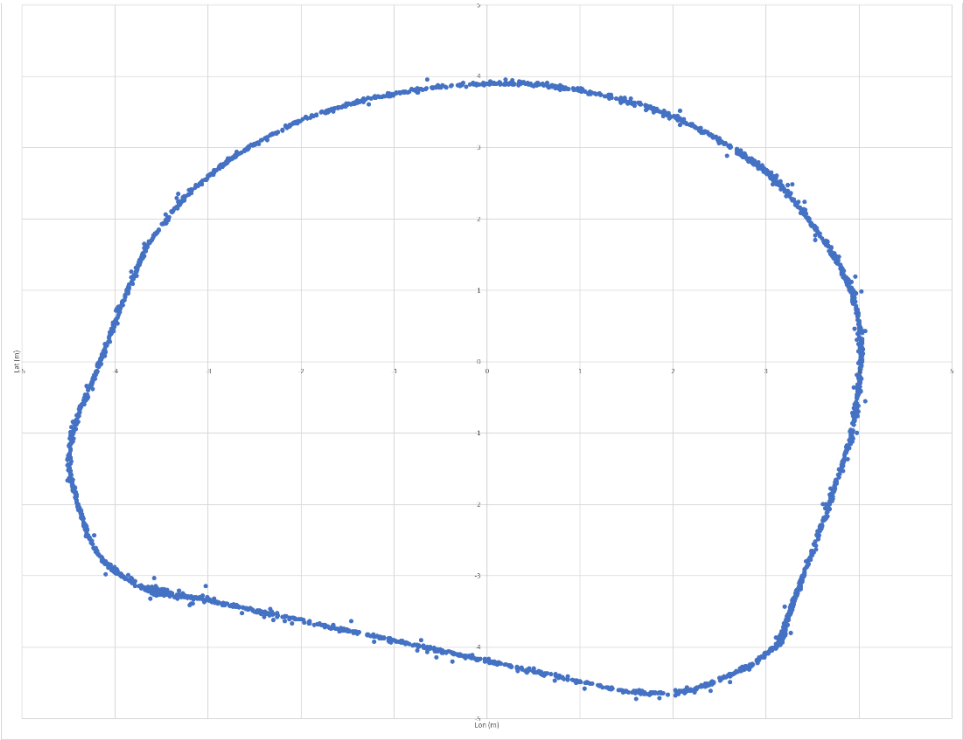


Figura 41. Medición del vehículo con la antena ANN-MB-00-00.

[14] Donde:

d	: distancia al punto de referencia
R	: radio de la tierra (6371 km)
lat	: latitud
long	: longitud

Al utilizar la antena AS-ANT2B-SUR-L1L2-25SMA-00, tal y como se aprecia en la Figura 42, el vehículo tiende a tener un menor número de desviaciones que con la antena anterior. Además, las desviaciones son menores ya que con esta antena la posición del vehículo no llega a desviarse más de 5 centímetros.

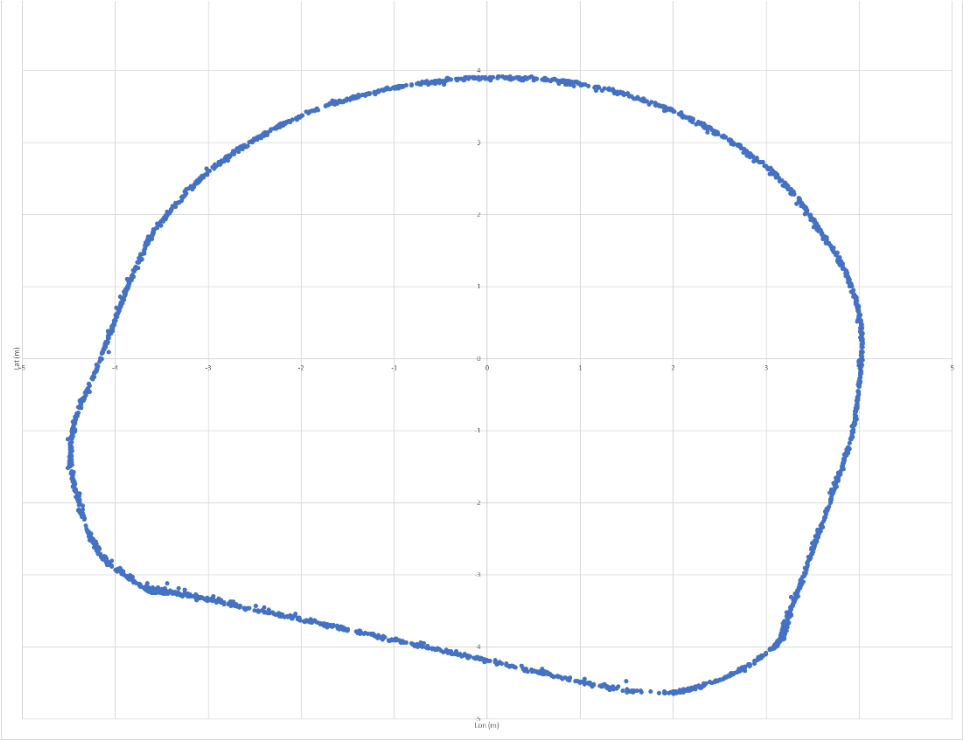


Figura 42. Medición de la posición con la antena AS-ANT2B-SUR-L1L2-25SMA-00.

En caso de desear más información, es posible acceder al documento de Excel compartido junto a la memoria. De todas formas, tras analizar los resultados se puede afirmar que la corrección del sistema de posicionamiento es suficiente como para que el vehículo pueda circular entre los cultivos de una plantación.

6 Líneas futuras

Tras analizar los resultados obtenidos, se puede ver que hay distintos frentes por los que se puede continuar el trabajo realizado.

Por un lado, se desea proseguir con lo descrito en la memoria realizando una comparación en cada momento del flujo de corrección enviado por radio y por protocolo NTRIP. De esta forma, se podría escoger la mejor transmisión en cada momento, garantizando que la corrección que se realice en el vehículo sea la mayor posible. Dependiendo así en menor medida de los obstáculos en el terreno que empeoran la señal de radio o de las pérdidas de la señal de Internet que dificultan la transmisión con el protocolo NTRIP.

Por otro lado, a pesar de que el caster creado logre transmitir la corrección entre un gran volumen de bases y clientes de manera simultánea y que pueda configurarse para solicitar un usuario y su correspondiente contraseña, no se ha podido habilitar el uso de usuarios en el servicio, al no poder transmitir esta información a través del ESP32. De modo que, se debe modificar el código del ESP32 para aumentar la seguridad del servicio y que no pueda conectarse cualquier cliente al caster.

Finalmente, si se desease trabajar en invernaderos, al tratarse de espacios cubiertos, la corrección en la posición, además de tener errores mayores que en espacios abiertos, se recibiría de manera entrecortada. Por lo que se tendría que encontrar una alternativa que permita recorrer al vehículo entre las hileras de cultivo. Una de las propuestas estudiadas consiste en utilizar la transformada de Hough para detectar las líneas de cultivo, permitiendo que el vehículo pueda circular entre las plantaciones. La transformada de Hough es una de las técnicas más utilizadas en la detección de líneas en imágenes y en el ámbito de los vehículos autónomos, se emplea comúnmente para la detección de las líneas de carril.

7 Conclusión

Tras analizar los resultados obtenidos se pueden llegar a las siguientes conclusiones:

1. Por razones tanto económicas como temporales, se recomienda emplear una base permanente en el caso que se disponga de alguna en el área de actuación, ya que, de esta forma, no sería necesario instalar una base propia.
2. La base debe ubicarse en un entorno abierto sin obstáculos en el terreno que puedan interferir con la señal satelital, por lo que se recomienda situarla en un emplazamiento con altura como puede ser un tejado.
3. Cuanto mayor sea el tiempo en el que se calcule la posición de la base menor será el error en su posición. Por lo que se ha visto, resulta recomendable hacer mediciones de 24 horas.
4. La transmisión del flujo de corrección a través del protocolo NTRIP depende en gran medida de la cobertura de la señal de Internet, mientras que la comunicación por radio se ve afectada por los obstáculos del terreno. Es necesario analizar la zona de cultivo para saber cuál es la comunicación más adecuada.

8 Bibliografía

[1] Cool Berries, «COMPAÑIA - Coolberries,» [En línea]. Available: https://coolberries.bio/compania/?lang=es. [Último acceso: Abril 2023].

[2] Cool Berries, «INNOVATION - Coolberries,» [En línea]. Available: https://coolberries.bio/innovation/. [Último acceso: Abril 2023].

[3] Gabri, «¿Cómo funcionan los dispositivos GPS? Trilateración Vs. Triangulación,» Mayo 2018. [En línea]. Available: https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/.

[4] D. G. Ariza, «Estudio de la calidad métrica de las observaciones en tiempo real en la Red GNSS de Castilla y León,» 2009. [En línea]. Available: https://gnss.itacyl.es/documents/1088492/0/Estudio_Calidad_metrica_RTK_en_Red_GNSS_de_CyL.pdf/9631a203-92ec-4385-8a65-61921b661900?t=1596707079261.

[5] E. M. Arias, «INTEGRACIÓN DE SISTEMA REAL-TIME KINEMATIC CON UNIDAD DE MEDICIÓN INERCIAL DENTRO DEL SOFTWARE AEROSTACK,» Febrero 2019. [En línea]. Available: https://oa.upm.es/54091/1/TFG_ELENA_MARTIN_ARIAS.pdf.

[6] Radio Technical Comission for Maritime Services, «Networked Transport of RTCM via Internet Protocol (NTRIP),» Julio 2018. [En línea]. Available: https://gssc.esa.int/wp-content/uploads/2018/07/NtripDocumentation.pdf.

[7] ArduSimple, «Centimeter Precision GPS/GNSS - RTK Explained,» [En línea]. Available: https://www.ardusimple.com/rtk-explained/. [Último acceso: 2023].

[8] Sparkfun, «How to Build a DIY GNSS Reference Station,» Octubre 2020. [En línea]. Available: https://learn.sparkfun.com/tutorials/how-to-build-a-diy-gnss-reference-station/all#mini-computer-setup-option-1.

[9] S. Choy, «GNSS Precise Point Positioning (PPP),» Enero 2018. [En línea]. Available: https://www.unoosa.org/documents/pdf/icg/2018/ait-gnss/16_PPP.pdf.

[10] A. Broekman, «DIY RTK GNSS: Part 2 – NTRIP Caster,» Mayo 2021. [En línea]. Available: https://www.youtube.com/watch?v=CRZlIM8PLsE&t=2s.

[11] rinex20, «NTRIPCASTER Docker,» Enero 2023. [En línea]. Available: https://hub.docker.com/r/rinex20/ntripcaster.

[12] J. E. S. Manrriquez, «Creación de contenedores Docker sobre Construcción de APIs REST a partir de Ontologías y Grafos de Conocimientos,» Junio 2021. [En línea]. Available: https://oa.upm.es/68543/1/TFG_JOSE_ELIAS_SILVA_MANRRIQUEZ.pdf.

[13] Bikram, «Docker Objects,» Septiembre 2021. [En línea]. Available: https://bikramat.medium.com/docker-objects-e561f0ce3365.

[14] Universitat Oberta de Catalunya, «Sistemas de comunicación en la banda ISM,» [En línea]. Available: https://openaccess.uoc.edu/bitstream/10609/141046/21/PLA4_Sistemas%20de%20comunicaci%C3%B3n%20en%20la%20banda%20ISM.pdf. [Último acceso: Mayo 2023].

[15] ¡Eureka! Technology, «Configuración XBee con XCTU 2021,» Agosto 2020. [En línea]. Available: https://www.youtube.com/watch?v=u4dEY20hrOY&t=306s.

[16] State Islamic University Sunan Gunung Jati Bandung, «Implementation of Haversine Formula for Counting Event Visitor in The Radius Based on Android Application,» [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7577575. [Último acceso: Mayo 2023].

9 Agradecimientos

Deseo expresar mi más sincero agradecimiento a los integrantes del proyecto Mula, Iban Latasa, Natxo Varona y Ander Ayesa, por su inestimable ayuda y paciencia durante mi periodo de prácticas en IED. Estoy realmente agradecido por la oportunidad de aprender y quiero que sepan que esto no hubiese sido posible sin su apoyo.

En particular, quiero agradecer a Ander Ayesa por la perseverancia al responder todas y cada una de mis preguntas acerca de AWS. Gracias a su dedicación, ahora tengo la posibilidad de adentrarme en el mundo de la computación en la nube.

Finalmente, quiero dar las gracias a Rafael Rogelio García por ser mi tutor en este proyecto. Su guía y conocimientos han sido invaluables a la hora de realizar este trabajo.

ANEXOS

Índice

Anexo I: Enlaces de descarga 57

Anexo II: Datos de la posición de la base 58

Anexo III: Programación de ESP32 para I2C 65

Anexo IV: Programación de ESP32 para Serial 70

Anexo V: Tarea para la ejecución del Docker en AWS (JSON) 75

Anexo I: Enlaces de descarga

- U-center
- RTKLIB
- SNIP
- RTK2GO mountpoint
- Arduino IDE
- Docker Desktop
- AWS CLI
- XCTU
- Mission Planner

Anexo II: Datos de la posición de la base

CSRS-PPP 3.54.2 (2022-11-10)

COM16___9600_230411_084501.obs

Data Start	Data End	Duration of Observations
2023-04-11 15:00:00.99	2023-04-12 06:03:56.00	15:03:55.010
Processing Time		**Product Type**
08:25:52 UTC 2023/04/12		NRCan Ultra-rapid
Observations	**Frequency**	**Mode**
Phase and Code	Double	Static

Elevation Cut-Off	Rejected Epochs	Fixed Ambiguities	Estimation Steps
7.5 degrees	84.11 %	0.00 %	1.00 sec

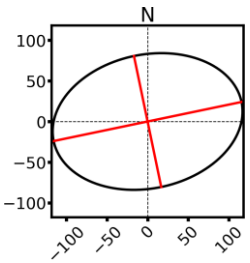
Antenna Model	APC to ARP	ARP to Marker
Unknown	Unknown	H:0.000m / E:0.000m / N:0.000m

(APC = antenna phase center; ARP = antenna reference point)

Estimated Position for COM16___9600_230411_084501.obs

	Latitude (+n)	Longitude (+e)	Ell. Height
ITRF20 (2023.3)	42° 50' 47.83851"	-1° 41' 5.60825"	488.876 m
Sigmas(95%)	0.673 m	0.935 m	1.962 m
A priori*	42° 50' 47.28393"	-1° 41' 12.55466"	19.293 m
Estimated – A priori	17.113 m	157.729 m	469.583 m

95% Error Ellipse (cm)
semi-major: 118.0 cm
semi-minor: 82.3 cm
semi-major azimuth: 78° 14' 25.05"

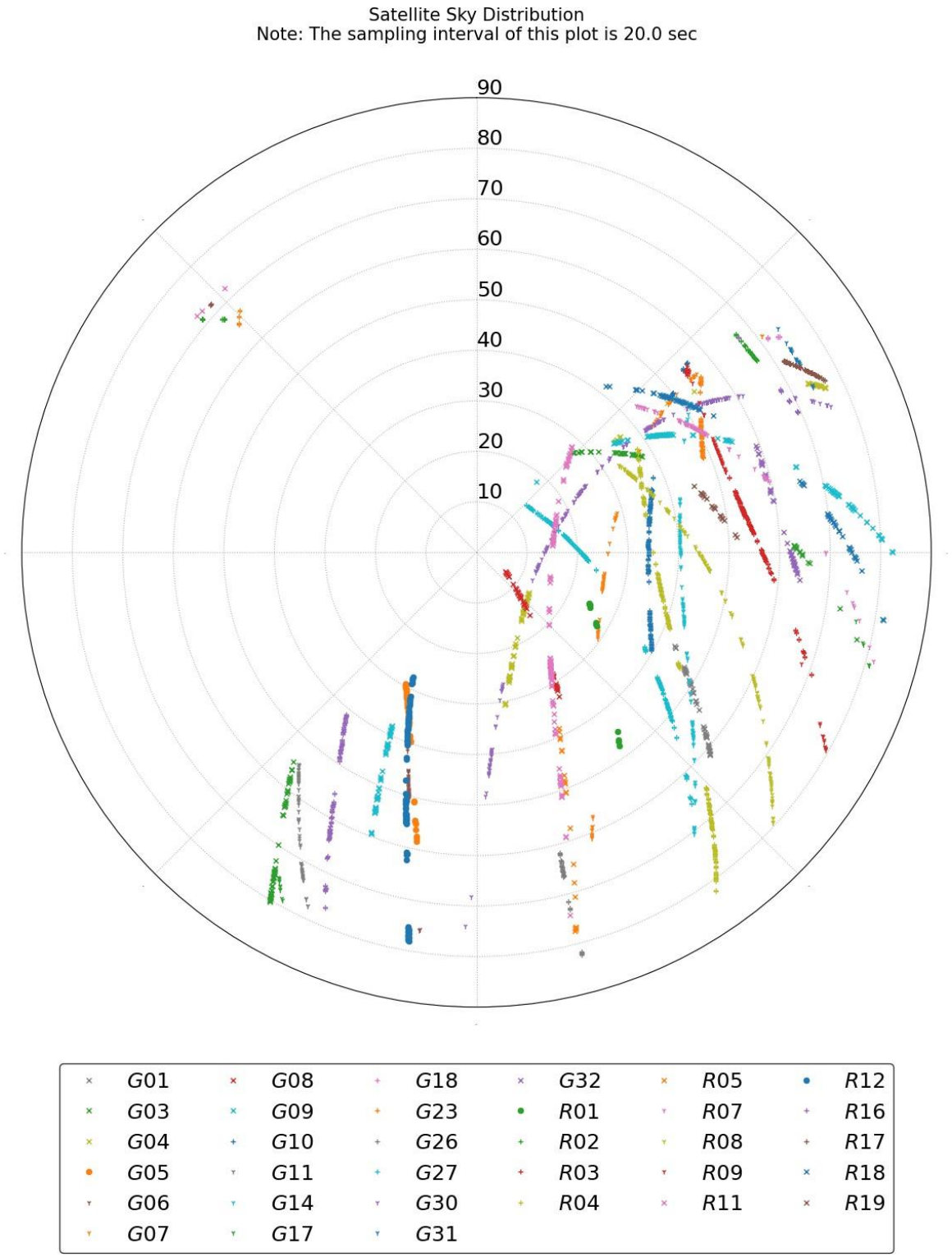


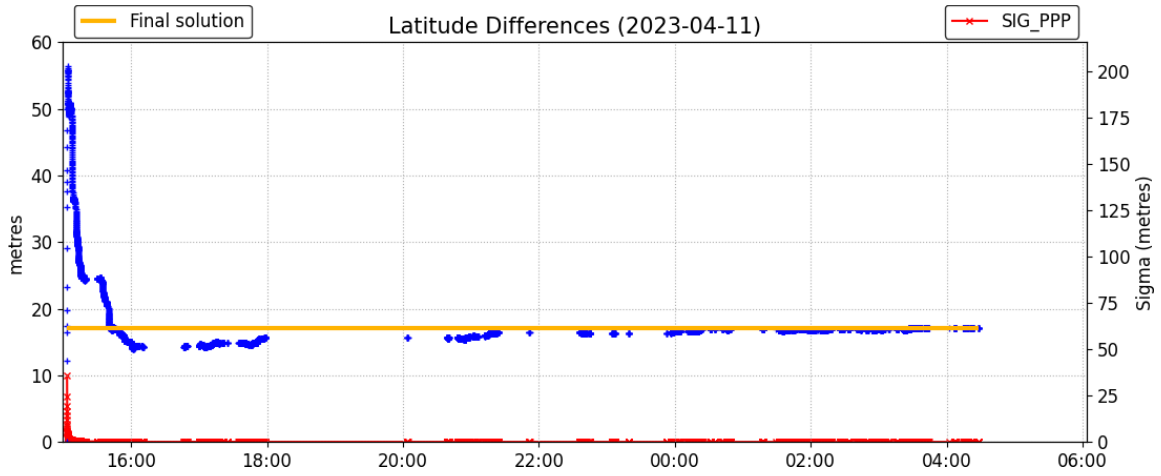
UTM (North)
Zone 30

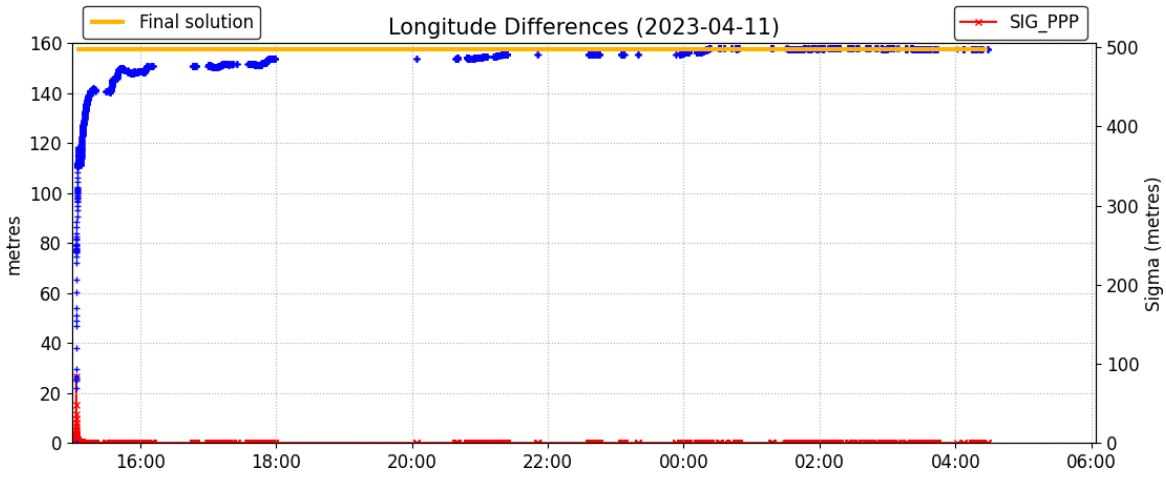
4744621.376 m (N)
607459.315 m (E)

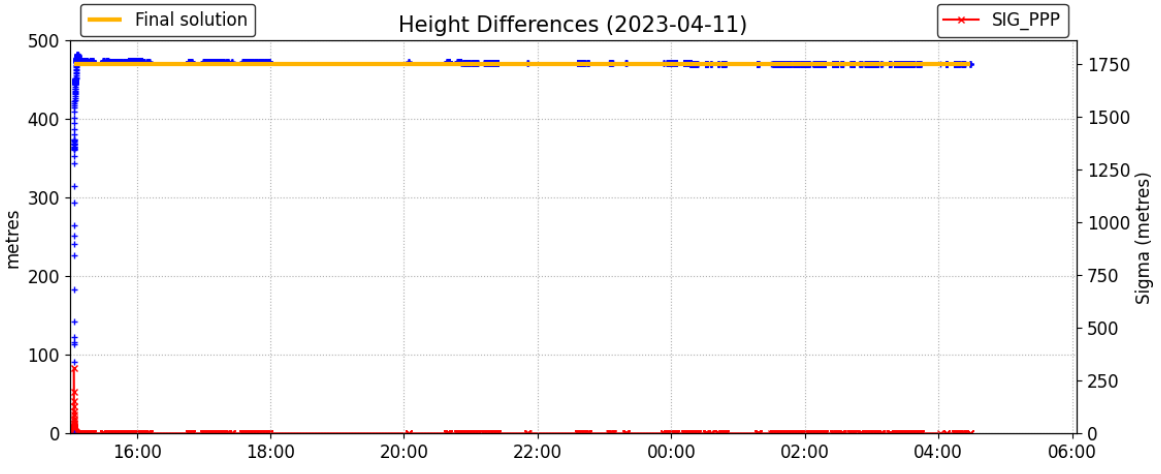
Scale Factors
0.99974206 (point)
0.99966542 (combined)

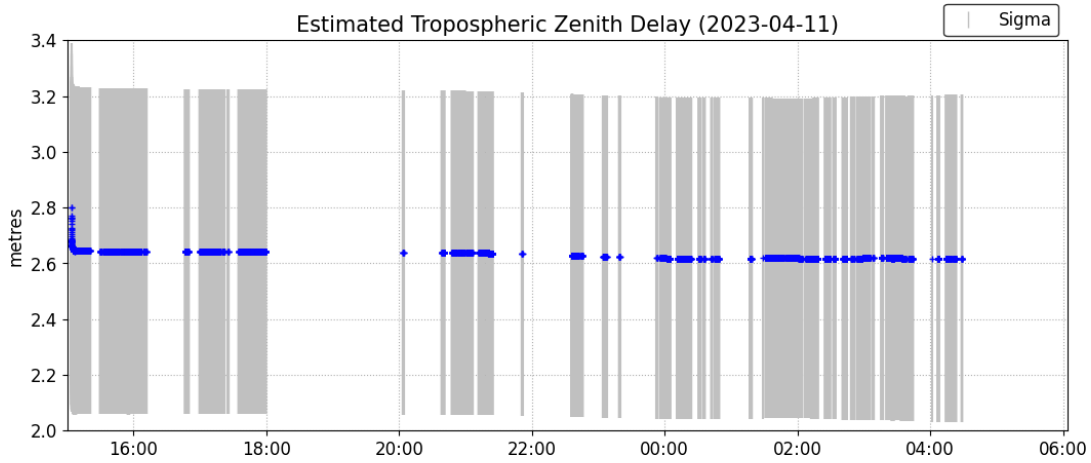
*(Coordinates from a code solution used as a priori position)

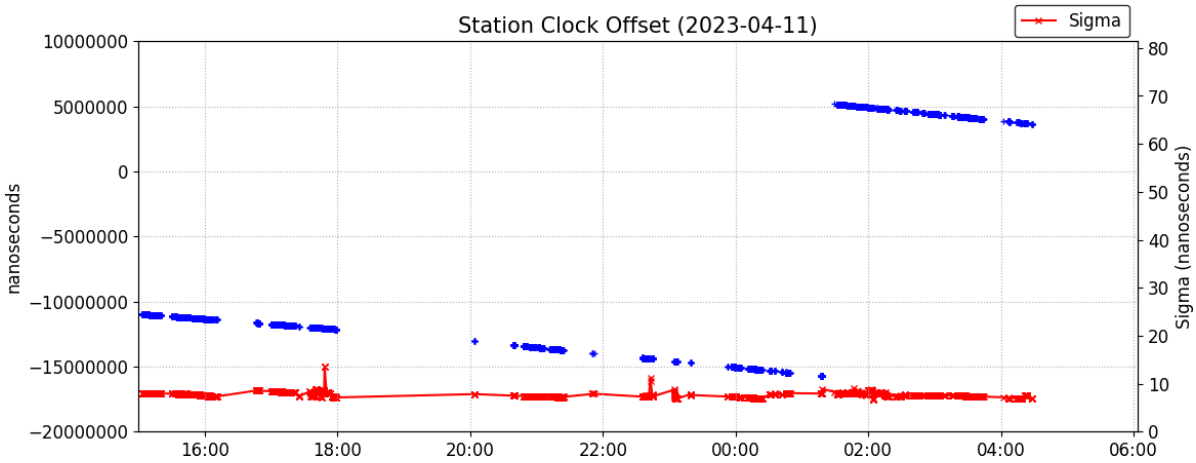

Satellite Sky Distribution
Note: The sampling interval of this plot is 20.0 sec
90
80
70
60
50
40
30
20
10
G01
G03
G04
G05
G06
G07
G08
G09
G10
G11
G14
G17
G18
G23
G26
G27
G30
G31
G32
R01
R02
R03
R04
R05
R07
R08
R09
R11
R12
R16
R17
R18
R19

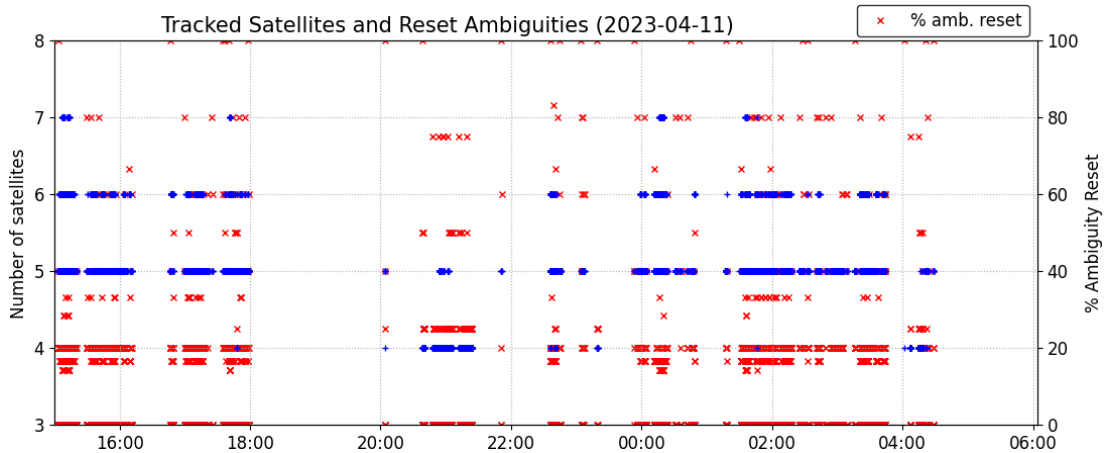

Final solution
Latitude Differences (2023-04-11)
SIG_PPP
metres
Sigma (metres)
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00

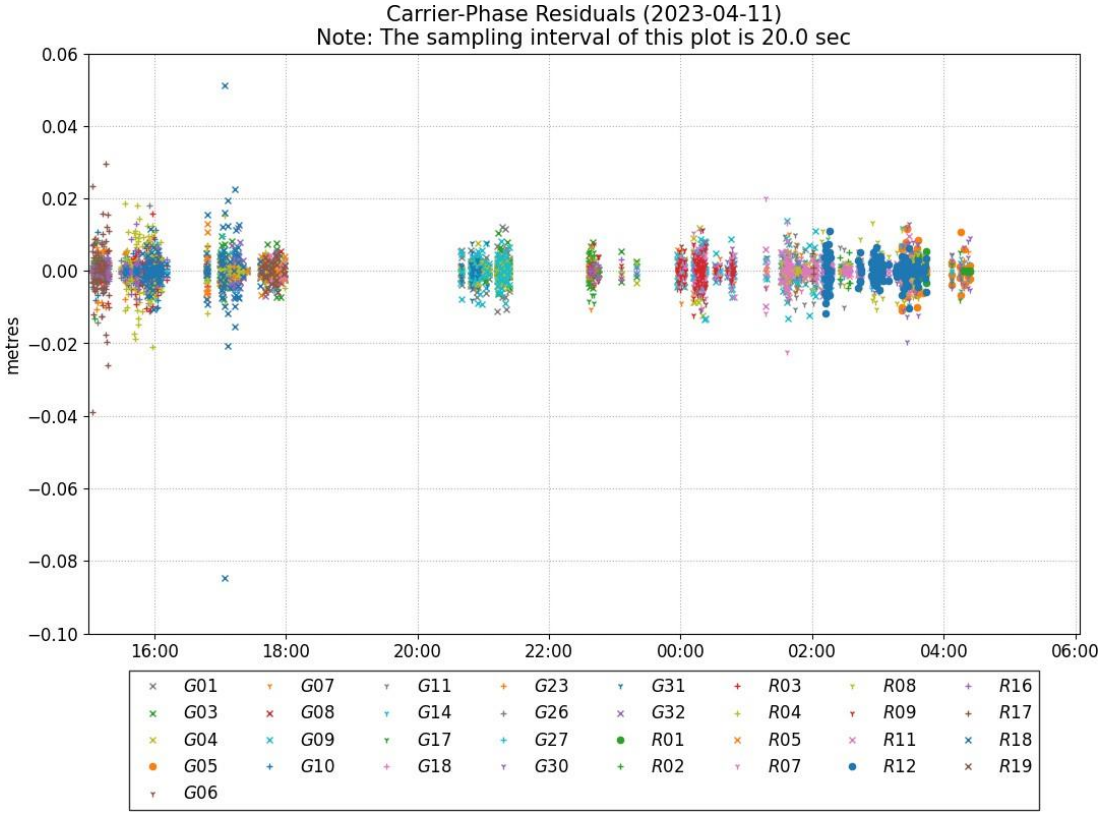

Final solution
Longitude Differences (2023-04-11)
SIG_PPP
metres
Sigma (metres)
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00

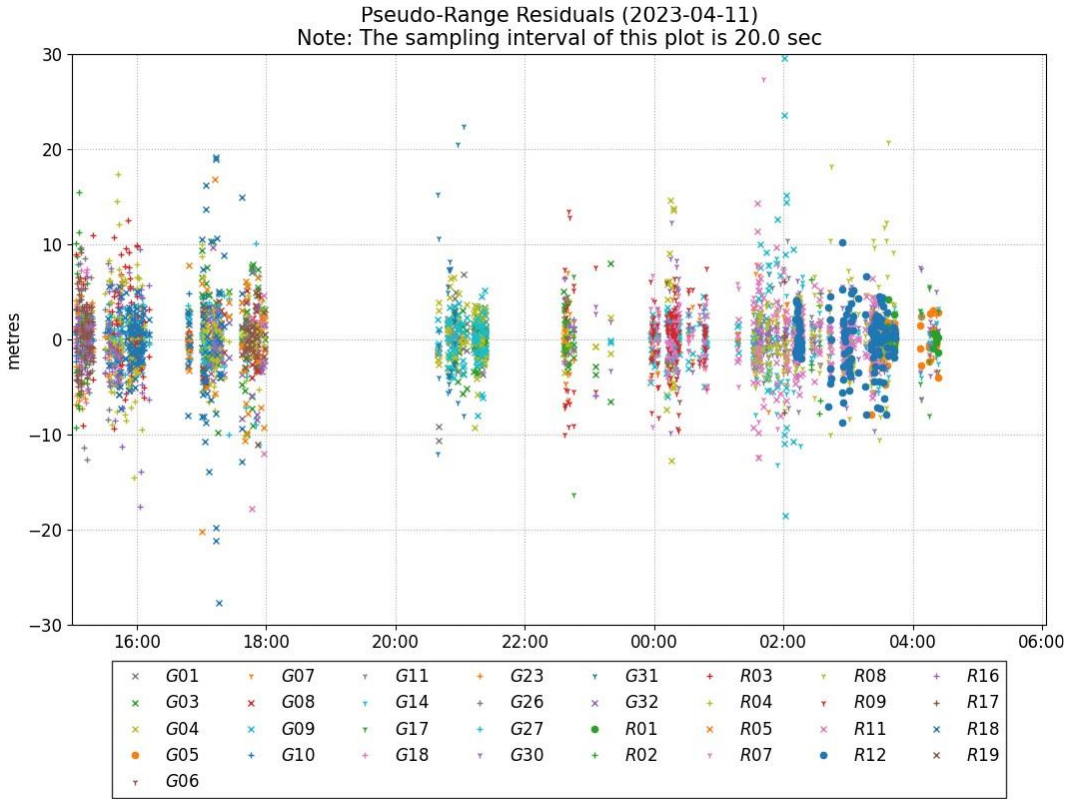

Final solution
Height Differences (2023-04-11)
SIG_PPP
metres
Sigma (metres)
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00

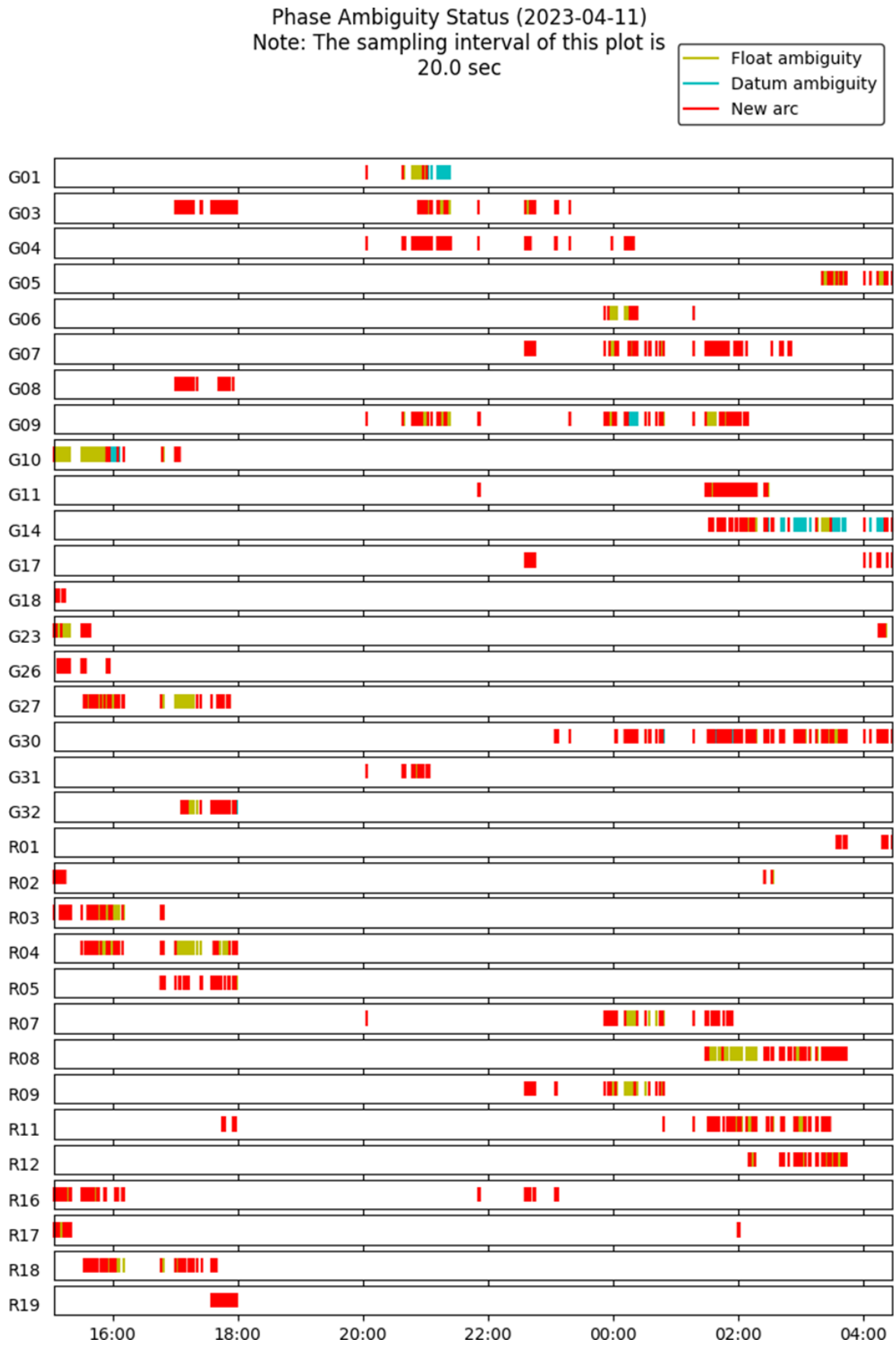

Estimated Tropospheric Zenith Delay (2023-04-11)
Sigma
metres
3.4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00


Station Clock Offset (2023-04-11)
Sigma
nanoseconds
Sigma (nanoseconds)
10000000
5000000
0
−5000000
−10000000
−15000000
−20000000
80
70
60
50
40
30
20
10
0
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00


Tracked Satellites and Reset Ambiguities (2023-04-11)
% amb. reset
Number of satellites
% Ambiguity Reset
8
7
6
5
4
3
100
80
60
40
20
0
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00


Carrier-Phase Residuals (2023-04-11)
Note: The sampling interval of this plot is 20.0 sec
metres
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
G01 G03 G04 G05 G06 G07 G08 G09 G10 G11 G14 G17 G18 G23 G26 G27 G30 G31 G32 R01 R02 R03 R04 R05 R07 R08 R09 R11 R12 R16 R17 R18 R19


Pseudo-Range Residuals (2023-04-11)
Note: The sampling interval of this plot is 20.0 sec
metres
30
20
10
0
−10
−20
−30
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
G01 G03 G04 G05 G06 G07 G08 G09 G10 G11 G14 G17 G18 G23 G26 G27 G30 G31 G32 R01 R02 R03 R04 R05 R07 R08 R09 R11 R12 R16 R17 R18 R19


Phase Ambiguity Status (2023-04-11)
Note: The sampling interval of this plot is 20.0 sec
Float ambiguity
Datum ambiguity
New arc
G01
G03
G04
G05
G06
G07
G08
G09
G10
G11
G14
G17
G18
G23
G26
G27
G30
G31
G32
R01
R02
R03
R04
R05
R07
R08
R09
R11
R12
R16
R17
R18
R19
16:00
18:00
20:00
22:00
00:00
02:00
04:00

~~~ Disclaimer ~~~

Natural Resources Canada does not assume any liability deemed to have been caused directly or indirectly by any content of its CSRS-PPP online positioning service.

If you have any questions, please feel free to contact:
Geodetic Integrated Services
Canadian Geodetic Survey
Surveyor General Branch
Natural Resources Canada
Government of Canada
588 Booth Street, Room 334
Ottawa, Ontario K1A 0Y7
Phone: 343-292-6617
Email: geodeticinformation-informationgeodesique@nrcan-rncan.gc.ca

Natural Resources Canada Ressources naturelles Canada Canada

Anexo III: Programación de ESP32 para I2C

```
//Libraries
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
#include <WiFi.h>             //This library allows WiFi communication
#include "secrets.h"          //This file includes important information for
the Caster (CasterHost, Mountpoint, SSID, ...)
WiFiClient ntripCaster;

#include <Wire.h>
#include <SparkFun_u-blox_GNSS_Arduino_Library.h>
//http://librarymanager/All#SparkFun_u-blox_GNSS
SFE_UBLOX_GNSS myGNSS;

//Global Variables
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
long lastSentRTCM_ms = 0;           //Time of last data pushed to socket
int maxTimeBeforeHangup_ms = 10000; //If we fail to get a complete RTCM frame
after 10s, then disconnect from caster

uint32_t serverBytesSent = 0; //Just a running total
long lastReport_ms = 0;       //Time of last report of bytes sent
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

//=============================================================================
//                    Power on setup
//=============================================================================
void setup()
{
  Serial.begin(115200);                    //Starts the serial communication
between the microcontroller and the serial monitor
  Serial.println(F("Initialing program"));

  Wire.begin();

  //myGNSS.enableDebugging(); // Uncomment this line to enable debug messages

  if (myGNSS.begin() == false) //Connect to the u-blox module using Wire port
  {
    Serial.println(F("u-blox GNSS not detected at default I2C address. Please
check wiring. Freezing."));
    while (1)
      ;
  }

  Serial.print("Connecting to local WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  Serial.print("\nWiFi connected with IP: ");
```

```
  Serial.println(WiFi.localIP());

  myGNSS.setI2COutput(COM_TYPE_UBX | COM_TYPE_NMEA | COM_TYPE_RTCM3);
//UBX+RTCM3 is not a valid option so we enable all three.
  //myGNSS.saveConfiguration();
  myGNSS.setNavigationFrequency(1); //Set output in Hz. RTCM rarely benefits
from >1Hz.

  //Disable all NMEA sentences
  bool response = true;
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_GGA, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_GSA, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_GSV, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_RMC, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_GST, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_GLL, COM_PORT_I2C);
  response &= myGNSS.disableNMEAMessage(UBX_NMEA_VTG, COM_PORT_I2C);

  if (response == false)
  {
    Serial.println(F("Failed to disable NMEA. Freezing..."));
    while (1)
      ;
  }
  else
    Serial.println(F("NMEA disabled"));

  //Enable necessary RTCM sentences
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1005, COM_PORT_I2C, 1);
//Enable message 1005 to output through I2C, message every second
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1074, COM_PORT_I2C, 1);
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1084, COM_PORT_I2C, 1);
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1094, COM_PORT_I2C, 1);
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1124, COM_PORT_I2C, 1);
  response &= myGNSS.enableRTCMmessage(UBX_RTCM_1230, COM_PORT_I2C, 10);
//Enable message every 10 seconds

  if (response == false)
  {
    Serial.println(F("Failed to enable RTCM. Freezing..."));
    while (1)
      ;
  }
  else
    Serial.println(F("RTCM sentences enabled"));

  myGNSS.setStaticPosition(468188940, 10, -13771846, -80, 431535712, 70);
//Gives base station's coordinates

  Serial.println(F("Static position set"));

  //If you were setting up a full GNSS station, you would want to save these
settings.
  //Because setting an incorrect static position will disable the ability to
get a lock, we will skip saving during this example
```

```
  //if (myGNSS.saveConfiguration() == false) //Save the current settings to flash and BBR
  //  Serial.println(F("Module failed to save"));

  Serial.println(F("Module configuration complete"));
}

void loop()
{

  Serial.println("Begin transmitting to caster");
  delay(10); //Wait for any serial to arrive


  while(true)
  {
    //Connect if we are not already
    if (ntripCaster.connected() == false)
    {
      Serial.printf("Opening socket to %s\n", casterHost);

      if (ntripCaster.connect(casterHost, casterPort) == true) //Attempt connection
      {
        Serial.printf("Connected to %s:%d\n", casterHost, casterPort);

        const int SERVER_BUFFER_SIZE = 512;
        char serverRequest[SERVER_BUFFER_SIZE];

        snprintf(serverRequest,
                 SERVER_BUFFER_SIZE,
                 "SOURCE %s /%s\r\nSource-Agent: NTRIP SparkFun u-blox Server v1.0\r\n\r\n",
                 mountPointPW, mountPoint);

        Serial.println(F("Sending server request:"));
        Serial.println(serverRequest);
        ntripCaster.write(serverRequest, strlen(serverRequest));

        //Wait for response
        unsigned long timeout = millis();
        while (ntripCaster.available() == 0)
        {
          if (millis() - timeout > 5000)
          {
            Serial.println("Caster timed out!");
            ntripCaster.stop();
            return;
          }
          delay(10);
        }

        //Check reply
        bool connectionSuccess = false;
        char response[512];
        int responseSpot = 0;
```

```
          while (ntripCaster.available())
165       {
            response[responseSpot++] = ntripCaster.read();
            if (strstr(response, "200") > 0) //Look for 'ICY 200 OK'
              connectionSuccess = true;
            if (responseSpot == 512 - 1)
170           break;
          }
          response[responseSpot] = '\0';

          if (connectionSuccess == false)
175       {
            Serial.printf("Failed to connect to Caster: %s", response);
            return;
          }
        } //End attempt to connect
180     else
        {
          Serial.println("Connection to host failed");
          return;
        }
185   } //End connected == false

      if (ntripCaster.connected() == true)
      {
        delay(10);
190     while (Serial.available())
          Serial.read(); //Flush any endlines or carriage returns

        lastReport_ms = millis();
        lastSentRTCM_ms = millis();
195
        //This is the main sending loop. We scan for new ublox data but
processRTCM() is where the data actually gets sent out.
        while (1)
        {
200       if (Serial.available())
            break;

          myGNSS.checkUblox(); //See if new data is available. Process bytes as
they come in.
205
          //Close socket if we don't have new data for 10s
          //RTK2Go will ban your IP address if you abuse it. See
http://www.rtk2go.com/how-to-get-your-ip-banned/
          //So let's not leave the socket open/hanging without data
210       if (millis() - lastSentRTCM_ms > maxTimeBeforeHangup_ms)
          {
            Serial.println("RTCM timeout. Disconnecting...");
            ntripCaster.stop();
            return;
215       }

          delay(10);

          //Report some statistics every 250
```

```
          if (millis() - lastReport_ms > 250)
          {
            lastReport_ms += 250;
            Serial.printf("Total sent: %d\n", serverBytesSent);
          }
        }
      }

      delay(10);
    }

    Serial.println("User pressed a key");
    Serial.println("Disconnecting...");
    ntripCaster.stop();

    delay(10);
    while (Serial.available())
      Serial.read(); //Flush any endlines or carriage returns
  }

  //This function gets called from the SparkFun u-blox Arduino Library.
  //As each RTCM byte comes in you can specify what to do with it
  //Useful for passing the RTCM correction data to a radio, Ntrip broadcaster,
  etc.
  void SFE_UBLOX_GNSS::processRTCM(uint8_t incoming)
  {
    if (ntripCaster.connected() == true)
    {
      ntripCaster.write(incoming); //Send this byte to socket
      serverBytesSent++;
      lastSentRTCM_ms = millis();
    }
  }
```

Anexo IV: Programación de ESP32 para Serial

```
//Libraries
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#include <WiFi.h>              //This library allows WiFi communication
#include "secrets.h"           //This file includes important information for the Caster (CasterHost, Mountpoint, SSID, ...)
WiFiClient ntripCaster;

#include <SoftwareSerial.h>    //This library allows serial communication between two digital pins
SoftwareSerial mySerial(1,2); //Pin 1 (RX) goes to TX pin on GNSS module and pin 3 (TX) goes to RX pin on GNSS module

#include <SparkFun_u-blox_GNSS_Arduino_Library.h> //Click here to get the library: http://librarymanager/All#SparkFun_u-blox_GNSS
SFE_UBLOX_GNSS myGNSS;

//Global Variables
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
long lastSentRTCM_ms = 0;           //Time of last data pushed to socket
int maxTimeBeforeHangup_ms = 10000; //If we fail to get a complete RTCM frame after 10s, then disconnect from caster

uint32_t serverBytesSent = 0; //Just a running total
long lastReport_ms = 0;       //Time of last report of bytes sent

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

//=========================================================================
//                      Power on setup
//=========================================================================
void setup()
{
  Serial.begin(115200);                     //Starts the serial communication between the microcontroller and the serial monitor
  Serial.println(F("Initialing program"));

  //myGNSS.enableDebugging(); // Uncomment this line to enable debug messages
  delay(1);
  do {
    Serial.println("GNSS: trying 9600 baud");
    mySerial.begin(9600);                         //Try a serial communication with the GNSS device
    if (myGNSS.begin(mySerial) == true) break;    //If the connection is established, it will exit the loop
    delay(1000);                                  //Wait 1 second before attempting to communicate again
    }
   while(1);
```

```
    Serial.print("Connecting to local WiFi");
    WiFi.begin(ssid, password);                    //Connects to the WiFi
network specified in secrets.h
    while (WiFi.status() != WL_CONNECTED)          //Enters a loop while the
connection status is not connected
    {
      delay(500);
      Serial.print(".");
    }

    Serial.print("\nWiFi connected with IP: ");
    Serial.println(WiFi.localIP());                //Prints local IP address
assigned to the device by the WiFi network

    myGNSS.setUART1Output(COM_TYPE_UBX | COM_TYPE_NMEA | COM_TYPE_RTCM3);
//Enables UBX, NMEA and RTCM3 messages

    myGNSS.setNavigationFrequency(1); //Set output in Hz. RTCM rarely benefits
from >1Hz.

    //Disable all NMEA sentences
    myGNSS.disableNMEAMessage(UBX_NMEA_GGA, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_GSA, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_GSV, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_RMC, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_GST, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_GLL, COM_PORT_UART1);
    myGNSS.disableNMEAMessage(UBX_NMEA_VTG, COM_PORT_UART1);

    Serial.println(F("NMEA disabled"));

    //Enable necessary RTCM sentences
    myGNSS.enableRTCMmessage(UBX_RTCM_1005, COM_PORT_UART1, 1); //Enable
message 1005 to output through UART1, message every second
    myGNSS.enableRTCMmessage(UBX_RTCM_1074, COM_PORT_UART1, 1);
    myGNSS.enableRTCMmessage(UBX_RTCM_1084, COM_PORT_UART1, 1);
    myGNSS.enableRTCMmessage(UBX_RTCM_1094, COM_PORT_UART1, 1);
    myGNSS.enableRTCMmessage(UBX_RTCM_1124, COM_PORT_UART1, 1);
    myGNSS.enableRTCMmessage(UBX_RTCM_1230, COM_PORT_UART1, 10); //Enable
message every 10 seconds

    Serial.println(F("RTCM sentences enabled"));

    myGNSS.setStaticPosition(468188940, 10, -13771846, -80, 431535712, 70);
//Gives base station's coordinates

  Serial.println(F("Static position set"));

    //If you were setting up a full GNSS station, you would want to save these
settings.
    //Because setting an incorrect static position will disable the ability to
get a lock, we will skip saving during this example
    //if (myGNSS.saveConfiguration() == false) //Save the current settings to
flash and BBR
    //  Serial.println(F("Module failed to save"));
```

```
    Serial.println(F("Module configuration complete"));
  }

void loop()
{

  Serial.println("Begin transmitting to caster");
  delay(10); //Wait for any serial to arrive


  while(true)
  {
    //Connect if we are not already
    if (ntripCaster.connected() == false)
    {
      Serial.printf("Opening socket to %s\n", casterHost);

      if (ntripCaster.connect(casterHost, casterPort) == true) //Attempt
connection
      {
        Serial.printf("Connected to %s:%d\n", casterHost, casterPort);

        const int SERVER_BUFFER_SIZE = 512;
        char serverRequest[SERVER_BUFFER_SIZE];

        snprintf(serverRequest,
                 SERVER_BUFFER_SIZE,
                 "SOURCE %s /%s\r\nSource-Agent: NTRIP SparkFun u-blox Server
v1.0\r\n\r\n",
                 mountPointPW, mountPoint);

        Serial.println(F("Sending server request:"));
        Serial.println(serverRequest);
        ntripCaster.write(serverRequest, strlen(serverRequest));

        //Wait for response
        unsigned long timeout = millis();
        while (ntripCaster.available() == 0)
        {
          if (millis() - timeout > 5000)
          {
            Serial.println("Caster timed out!");
            ntripCaster.stop();
            return;
          }
          delay(10);
        }

        //Check reply
        bool connectionSuccess = false;
        char response[512];
        int responseSpot = 0;
        while (ntripCaster.available())
        {
          response[responseSpot++] = ntripCaster.read();
          if (strstr(response, "200") > 0) //Look for 'ICY 200 OK'
```

```
            connectionSuccess = true;
          if (responseSpot == 512 - 1)
            break;
        }
        response[responseSpot] = '\0';

        if (connectionSuccess == false)
        {
          Serial.printf("Failed to connect to Caster: %s", response);
          return;
        }
      } //End attempt to connect
      else
      {
        Serial.println("Connection to host failed");
        return;
      }
    } //End connected == false

    if (ntripCaster.connected() == true)
    {
      delay(10);
      while (Serial.available())
        Serial.read(); //Flush any endlines or carriage returns

      lastReport_ms = millis();
      lastSentRTCM_ms = millis();

      //This is the main sending loop. We scan for new ublox data but processRTCM() is where the data actually gets sent out.
      while (1)
      {
        if (Serial.available())
          break;

        myGNSS.checkUblox(); //See if new data is available. Process bytes as they come in.

        //Close socket if we don't have new data for 10s
        //RTK2Go will ban your IP address if you abuse it. See http://www.rtk2go.com/how-to-get-your-ip-banned/
        //So let's not leave the socket open/hanging without data
        if (millis() - lastSentRTCM_ms > maxTimeBeforeHangup_ms)
        {
          Serial.println("RTCM timeout. Disconnecting...");
          ntripCaster.stop();
          return;
        }

        delay(10);

        //Report some statistics every 250
        if (millis() - lastReport_ms > 250)
        {
          lastReport_ms += 250;
          Serial.printf("Total sent: %d\n", serverBytesSent);
```

```
        }
      }
    }

    delay(10);
  }

  Serial.println("User pressed a key");
  Serial.println("Disconnecting...");
  ntripCaster.stop();

  delay(10);
  while (Serial.available())
    Serial.read(); //Flush any endlines or carriage returns
}

//This function gets called from the SparkFun u-blox Arduino Library.
//As each RTCM byte comes in you can specify what to do with it
//Useful for passing the RTCM correction data to a radio, Ntrip broadcaster,
etc.
void SFE_UBLOX_GNSS::processRTCM(uint8_t incoming)
{
  if (ntripCaster.connected() == true)
  {
    ntripCaster.write(incoming); //Send this byte to socket
    serverBytesSent++;
    lastSentRTCM_ms = millis();
  }
}
```

Anexo V: Tarea para ejecución del Docker en AWS (JSON)


```
{
    "taskDefinitionArn": "arn:aws:ecs:eu-central-1:435796944442:task-
definition/rtk-caster-task-micro:1",
    "containerDefinitions": [
        {
            "name": "rtk-caster-container",
            "image": "public.ecr.aws/k7l4c9u4/rtk-caster",
            "cpu": 0,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                },
                {
                    "containerPort": 2101,
                    "hostPort": 2101,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [],
            "command": [],
            "environment": [],
            "mountPoints": [
                {
                    "sourceVolume": "casterConfig",
                    "containerPath": "/etc/ntripcaster",
                    "readOnly": false
                }
            ],
            "volumesFrom": [],
            "interactive": true
        }
    ],
    "family": "rtk-caster-task-micro",
    "revision": 1,
    "volumes": [
        {
            "name": "casterConfig",
            "host": {
                "sourcePath": "/etc/ntripcaster"
            }
        }
    ],
    "status": "ACTIVE",
    "requiresAttributes": [
        {
            "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
        }
    ],
    "placementConstraints": [],
    "compatibilities": [
        "EXTERNAL",
        "EC2"
    ],
```

```
        "requiresCompatibilities": [
            "EC2"
        ],
        "cpu": "512",
        "memory": "500",
        "registeredAt": "2023-04-27T07:37:32.561Z",
        "registeredBy": "arn:aws:iam::435796944442:user/aayesa@iedcompany.com",
        "tags": []
    }
```