

HARDWARE

Base de tiempos precisa basada en GPS USB

Sergio Díaz Ruiz¹¹Asociación Astronomía Sevilla, Spain. E-mail: sergio.diaz.ruiz@gmail.com.**Keywords:** GPS, PPS, NTP, USB, gpsd, chrony, ntpd

© Este artículo está protegido bajo una licencia Creative Commons Attribution 4.0 License

Resumen

La observación de ocultaciones estelares, campo en el que la astronomía amateur supone una colaboración clave, requiere una resolución temporal del orden del milisegundo, siendo la señal GPS la mejor referencia temporal disponible. Este trabajo aborda una implementación práctica y económica que utiliza un módulo receptor GPS de bajo coste conectado mediante un conversor USB-serie, transportando la señal Pulse-Per-Second (PPS) a través de una de las líneas de control del interfaz serie. Esta solución, que ha sido recurrentemente descartada debido a las dudas en torno al impacto en la precisión temporal del transporte USB, se documenta aquí detalladamente, incluyendo la configuración hardware y software para entornos Linux (gpsd y chrony) y Windows (Meinberg NTP). Se destaca la importancia de un proceso de calibración de desfase, crucial para corregir las latencias inherentes a las capas de software intermedias. Los resultados de las pruebas confirman la viabilidad de la solución, demostrando que, tras la corrección del desfase, se logra una precisión por debajo del milisegundo en ambos sistemas.

Abstract

The observation of stellar occultations, a field in which amateur astronomy plays a key collaborative role, requires temporal resolution on the order of milliseconds, with the GPS signal being the best available time reference. This work addresses a practical and economical implementation that uses a low-cost GPS receiver module connected via a USB-to-serial converter, transporting the Pulse-Per-Second (PPS) signal through one of the control lines of the serial interface. This solution, which has been repeatedly dismissed due to doubts about the impact of USB transport on temporal accuracy, is documented here in detail, including the hardware and software configuration for Linux (gpsd and chrony) and Windows (Meinberg NTP) environments. The importance of an offset calibration process, crucial for correcting the latencies inherent in intermediate software layers, is highlighted. Test results confirm the viability of the solution, demonstrating that, after offset correction, sub-millisecond accuracy is achieved on both systems.

1. Introducción

La ciencia ciudadana posee una larga tradición en las ciencias naturales, siendo la práctica habitual hasta la especialización de las disciplinas científicas a finales del siglo XVIII. En astronomía, la aportación de los astrónomos amateur, especialmente en el campo de la observación de ocultaciones estelares [2], inicialmente por la Luna y más recientemente por asteroides y otros cuerpos del sistema solar, es especialmente destacable, dada la importancia que tiene contar con múltiples observaciones del mismo fenómeno, realizadas desde distintos puntos de la geografía terrestre [3].

Estas observaciones permiten conocer la forma de los cuerpos (y si presentan anillos) [3], estudiar su atmósfera [4] o determinar de forma precisa su posición, información que resulta crítica para misiones como, p.ej., NASA Lucy [5].

Para aportar datos de calidad a una campaña de ocultación de la IOTA (*International Occultation Timing Association*), es necesario que los tiempos de desaparición y reaparición se proporcionen con precisión suficiente, actualmente del orden de decenas de milisegundos o incluso mejor. Para ello es necesario disponer de una base de tiempos basada en GPS [3] para sincronizar el reloj del equipo de captura. En este contexto, nos referiremos al *equipo de captura* como el ordenador, normalmente portátil, con el que se realiza la captura de vídeo a alta tasa de fotogramas o *frames* usando una cámara CMOS o de vídeo analógico y su correspondiente dispositivo digitalizador.

El presente trabajo propone una solución adicional a las ya usadas habitualmente en este ámbito. Consiste en conectar un módulo receptor GPS de bajo coste con interfaz serie UART y señal PPS a través de un conversor USB-serie, transportando la señal PPS a través de una de las líneas de control. Se proporcionan los pasos necesarios tanto para su implementación hardware como la instalación y configuración del software necesario tanto en entornos Linux como Windows. Seguidamente se detalla el procedimiento de calibración del desfase y se analizan los resultados de las pruebas realizadas para evaluar la precisión de la solución.

Los scripts desarrollados para analizar el ajuste necesario para la calibración se encuentran disponibles en el repositorio GitHub [1].

2. Sincronización de tiempo

La sincronización de reloj es un problema de ingeniería enfocado en la coordinación de los relojes de distintos sistemas. En el ámbito que nos ocupa, no sólo es relevante que los relojes de los observadores estén coordinados entre sí, sino además que lo estén respecto de una referencia estándar absoluta como es UTC (*Coordinated Universal Time*), que permita relacionar las observaciones realizadas con las efemérides de los cuerpos observados.

En el ámbito de las redes informáticas, la sincronización de tiempo se implementa mediante protocolos de comunicaciones especialmente diseñados para tal fin, tales como NTP y PTP. NTP (*Network Time Protocol*) está ampliamente extendido, contando con implementaciones para esencialmente todos los sistemas operativos existentes. Habitualmente se implementa bajo una arquitectura cliente-servidor, de forma que varios servidores actúan como referencia de tiempo que cada cliente consulta para derivar una estimación de la hora, habitualmente con una exactitud del orden de milisegundos, más que suficiente para muchas aplicaciones.

PTP (*Precision Time Protocol*) busca una mayor exactitud, del orden de nanosegundos o incluso mejor, necesaria para aplicaciones críticas (sistemas financieros, automatización industrial, operación de redes 5G, proyectos científicos, etc.), mediante el uso de marcas de tiempo insertadas por hardware directamente en las tarjetas de red (NIC, *Network Interface Card*), que además incorporan osciladores de alta precisión. Actualmente, esta funcionalidad sólo está implementada en tarjetas de red para servidores y estaciones de trabajo, no estando presente en las tarjetas de red usadas en los equipos de captura, por lo que no es viable en nuestro caso.

En el ámbito que nos ocupa, que conlleva la realización de observaciones en campo a lo largo de la banda de sombra de la ocultación, en general no tendremos asegurada la conectividad de red. Por ello, la solución más sencilla y económica es usar un receptor GPS, que además presenta mayor exactitud temporal que la conseguida generalmente con NTP. Sin embargo, tanto para la gestión de la sincronización horaria en el ordenador, como para el proceso de calibración de desfase que debemos realizar sobre la hora obtenida por el GPS y que describimos más adelante en este artículo, necesitamos tener desplegado un servicio de hora basado en NTP en el equipo.

2.1. Conceptos fundamentales de NTP

NTP define tanto un protocolo de comunicaciones, orientado a capturar marcas de tiempo tanto del cliente como del servidor, como los algoritmos necesarios para derivar los parámetros de ajuste del reloj local del cliente para su sincronización con el del servidor.

El intercambio de mensajes NTP entre cliente y servidor, así como el fundamento básico para derivar a partir de los mismos tanto el desfase (*offset*) entre sus relojes como el retardo de ida y vuelta (*round-trip delay* o simplemente *delay*) debido a la red, se resume a continuación de manera muy simplificada [6][7][8].

El cliente envía un mensaje NTP insertando la marca de tiempo (*timestamp*) correspondiente al instante de envío, t_0 , basada en su reloj local, figura 1. El mensaje se recibe en el servidor en el instante t_1 según su propio reloj. En el caso ideal en que ambos relojes ya estuviesen sincronizados, la diferencia $t_1 - t_0$ correspondería a la latencia introducida por la red de comunicaciones, λ . Sin embargo, en la práctica, ambos relojes presentarán un desfase θ , de modo que $t_1 - t_0 = \lambda + \theta$. Aquí, θ se considera positivo si el reloj del servidor adelanta al del cliente y negativo en caso contrario. Tras procesar el mensaje, el servidor responde enviando un nuevo mensaje, en el que inserta el *timestamp* t_2 , que el cliente recibirá en t_3 . Suponiendo que la red presenta simetría, es decir, la latencia de red para este camino de vuelta es igual que para el de ida, y que el desfase entre los relojes se mantiene constante durante este intercambio de mensajes, tendremos que $t_3 - t_2 = \lambda - \theta$, ya que ahora el desfase afecta en sentido opuesto.

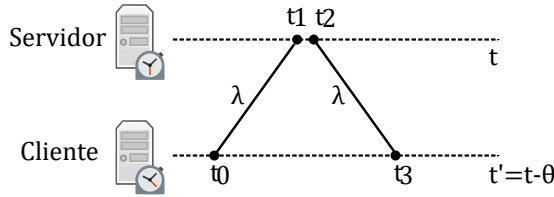


Figura 1. Esquema de envío de mensajes NTP entre cliente y servidor.

Sumando ambas ecuaciones obtenemos la latencia de red, $2\lambda = (t_1 - t_0) + (t_3 - t_2)$. Restándolas podemos aislar el desfase, $2\theta = (t_1 - t_0) - (t_3 - t_2)$. Dado que, en lugar de la latencia λ , habitualmente se suele usar el retardo de ida y vuelta (*round-trip delay*), δ , que en caso simétrico es $\delta = 2\lambda$, quedaría:

$$\delta = (t_1 - t_0) + (t_3 - t_2)$$

$$\theta = \frac{(t_1 - t_0) - (t_3 - t_2)}{2}$$

El cliente mide estos parámetros por **sondeo periódico** sobre todos los servidores que tenga configurados. Se aplica un algoritmo de **selección** que descarta las fuentes que no se consideran lo suficientemente fiables, *false tickers*. Sobre las fuentes supervivientes, *true chimers*, o el subconjunto más representativo de las mismas, se usa un algoritmo de **combinación** que trata de obtener la mejor estimación del desfase a aplicar sobre el reloj local. Este ajuste generalmente se realiza de forma gradual, proceso que se conoce como **disciplinar** el reloj. Todo el mecanismo descrito conforma un bucle de control retroalimentado que idealmente debe converger, coordinando el reloj del cliente con los servidores.

Por tanto, en un equipo dotado de un servicio NTP, cabe distinguir entre los siguientes tipos de relojes:

- **Reloj NTP:** reloj virtual, interno al servicio NTP, basado en el reloj del sistema y las fuentes externas de tiempo configuradas. Mantiene la mejor estimación posible de la hora real.

- **Reloj del sistema:** reloj software mantenido por el núcleo (*kernel*) del sistema operativo, en base al oscilador del sistema. Es el reloj que “ven” las aplicaciones. En equipos que dispongan del servicio NTP, este reloj es ajustado gradualmente (“*disciplinado*”) a partir de la hora del reloj NTP.
- **Reloj en tiempo real (RTC):** reloj de hardware que mantiene la hora cuando el ordenador está apagado. Permite inicializar el reloj del sistema en el arranque.

NTP define un sistema jerarquizado de fuentes de tiempo. Cada nivel se conoce con el nombre de *stratum*. El **stratum 0** corresponde a fuentes de tiempo tales como relojes atómicos, receptores GPS, relojes de radio, etc. Este tipo de fuente también recibe el nombre de **reloj de referencia**, *refclock*. Los servidores NTP conectados directamente a los relojes de referencia pertenecen al **stratum 1**. Los servidores NTP que se sincronizan a través de la red con servidores NTP *stratum 1* pertenecen al **stratum 2**, y así sucesivamente.

En este punto conviene aclarar que un ordenador que tenga desplegado un servicio NTP no tiene por qué desempeñar el rol de servidor NTP. Es decir, este servicio permite al ordenador también funcionar simplemente como cliente NTP.

2.2. Implementaciones de NTP

Actualmente existen varias implementaciones de NTP. La más destacable, por ser la desarrollada por la *Network Time Foundation*, entidad a cargo de las revisiones del propio protocolo, se denomina **ntpd**¹, siendo la implementación de referencia. Está disponible virtualmente en todos los sistemas operativos de uso general. En particular, es la única disponible en Windows, de la mano del fabricante *Meinberg*, que se encarga de compilar los fuentes y de liberar un instalador [22].

Otra implementación, más reciente y de uso muy extendido, es **chrony**, que incorpora algunas mejoras respecto de la implementación de referencia, si bien está centrada principalmente en sistemas Linux. A continuación se resumen algunas de sus características principales:

- **chrony** se adapta con mayor agilidad a cambios rápidos de frecuencia de reloj que **ntpd** [9].
- Al estar continuamente monitorizando la deriva de la frecuencia de reloj del sistema respecto del reloj NTP (*frequency tracking*) [21], **chrony** puede extrapolar las correcciones necesarias en entornos donde el acceso a la referencia de tiempo es intermitente. En estos casos **ntpd** no resulta efectivo ya que requiere un sondeo regular, y una vez recuperada la conectividad con la referencia, el ajuste de hora puede llevar bastante tiempo [9].
- De la misma forma, **chrony** también mide la deriva del reloj RTC, por lo que tras el arranque del sistema, la hora se puede ajustar en base a una versión corregida del reloj RTC [21][9].
- El algoritmo de combinación de **chrony** tiene en cuenta no sólo las medidas de desfase de las fuentes *truechimers*, sino también las medidas de frecuencia del reloj de sistema [21]. La disciplina de reloj se aplica independientemente tanto en fase como en frecuencia [9].

En el ámbito de este estudio, donde habitualmente el equipo de captura no está siempre encendido y con conexión a la referencia horaria, estas características hacen que **chrony** sea especialmente interesante, ya que tanto el ajuste de deriva del reloj RTC durante el arranque como el ajuste de deriva del reloj del sistema permiten una puesta en marcha de la sincronización de forma más ágil así como mayor estabilidad en caso de pérdida de conectividad de red o señal GPS. Sin embargo, como se ha mencionado anteriormente, sólo es aplicable a Linux. En el caso de Windows, la única opción es recurrir a la implementación de **ntpd** de *Meinberg*.

¹Si bien el proyecto se denomina **ntp**, en este documento nos referiremos a esta implementación con el nombre del programa, **ntpd**, para evitar confusión con el nombre del protocolo.

3. Sistemas de navegación por satélite

El Sistema Global de Navegación por Satélite (GNSS, *Global Navigation Satellite System*) es el término genérico para tecnologías que usan redes de satélites para determinar la posición, velocidad y tiempo en cualquier lugar del mundo [10]. Aunque en el uso popular los conocemos como "GPS", esta denominación corresponde específicamente al GNSS desplegado por EE.UU., siendo también GNSSs los sistemas Galileo (Europa), GLONASS (Rusia) y BeiDou (China). Aquí usaremos "GPS" de forma genérica para referirnos a cualquier GNSS.

En nuestro ámbito es de gran interés, no sólo por informar sobre la posición con una incertidumbre de tan sólo 2-10 metros, algo de gran interés para la observación de ocultaciones, sino por proporcionar un reloj de referencia UTC de gran precisión, del orden de decenas de nanosegundos.

3.1. Conceptos fundamentales

La radionavegación por satélite se basa en el cálculo de una posición sobre la superficie terrestre midiendo las distancias de un mínimo de tres satélites de posición conocida ("2D fix"). Un cuarto satélite aporta información suficiente para determinar también la altitud ("3D fix"). El receptor GPS capta los mensajes de sincronización emitidos por los satélites, que contiene la posición del satélite y la marca de tiempo en que fue enviado [10].

La precisión de la posición depende de la exactitud de la información de tiempo. Solo los relojes atómicos proveen la precisión requerida, del orden de nanosegundos, por ello son los que se usan a bordo de los satélites GPS. El receptor compara la hora codificada en la transmisión con el instante de la recepción medido por un reloj interno, de forma que se mide el *tiempo de vuelo* de la señal desde el satélite [10].

3.2. Implementaciones hardware para posicionamiento

Los receptores GPS presentan dos rangos de exactitud: el estándar, 2-10 metros, y el de alta precisión, 1-2 centímetros. Aunque existen módulos especializados en sincronización horaria, que emplean tecnologías como SBAS (Satellite-Based Augmentation System) para compensar el retardo ionosférico (usando estaciones en tierra para monitorizarlo) [11], no se consideran adecuados para este caso de uso, ya que estos módulos son en torno a diez veces más caros y la precisión adicional que ofrecen no sería aprovechada por los servicios basados en NTP.

El receptor GPS seleccionado para este proyecto es un módulo de bajo coste, Beitian BE-280, basado en chip u-blox M10050, y dispone de antena integrada, presentando un formato muy compacto (28 x 28 x 8.4 mm), figura 2. Es compatible con GPS, Galileo y BeiDou (BDS). Tiene una precisión de hasta 2 metros en posicionamiento y de 30 ns en tiempo [16].



Figura 2. Módulo Beitian BE-280 usado en las pruebas.

3.3. Interfaz de comunicaciones

Los receptores GPS suelen integrar un interfaz UART (*Universal Asynchronous Receiver Transmitter*) de tipo TTL para comunicarse con un ordenador. Los mensajes se envían secuencialmente bit a bit, a una velocidad configurable (expresada en bits por segundo, *bps*), agrupados por bytes. A cada byte le precede una marca de inicio, y le sucede una marca de fin. "TTL" hace referencia a los niveles de tensión usados para codificar los bits, en este caso 0 y 5 voltios. El interfaz dispone de líneas de comunicación separadas para recepción y transmisión de datos, denominadas RX y TX. Estrictamente, el receptor GPS sólo necesitaría la línea de transmisión, sin embargo, es frecuente usar ambas ya que eso permite al ordenador enviar comandos al receptor GPS, generalmente para ajustar la configuración del mismo. También son frecuentes interfaces I²C (*Inter-Integrated Circuit*), SPI (*Serial Periphere Interface*) e incluso CAN (*Controller Area Network*), si bien estos están orientados a sistemas empotrados e industriales.

Los mensajes enviados por el receptor GPS suelen estar en formato NMEA 0183. Se trata de un protocolo que define mensajes en texto plano, de entre los que cabe destacar GPRMC (*GPS - Recommended Minimum Specific Data*), que incorpora la información sobre posición, velocidad y tiempo. Algunos fabricantes incorporan protocolos propietarios, como es el caso de u-blox, que define el protocolo binario UBX. Al ser binario, permite mayor densidad de información: el equivalente a GPRMC sería UBX-NAV-PVT, que ofrece además una estimación del error de la geolocalización, el vector de velocidad y su incertidumbre, etc.

En cualquier caso, de cara a ofrecer una referencia horaria, se plantea el problema de que la marca horaria enviada en el mensaje al ordenador tarda un cierto tiempo en recibirse, que además depende también del tiempo que tarde el receptor en calcular la solución de geolocalización y tiempo. Esto añade un retardo considerable y lo que es peor, una incertidumbre respecto a qué instante, medido con el reloj de sistema, correspondía a la marca temporal recibida en el mensaje.

Por este motivo, los receptores GPS suelen incorporar una línea de salida adicional, denominada *Pulse-Per-Second*, PPS ó 1PPS, por la que envían un pulso de 100 ms (típ.) cuyo flanco ascendente coincide con el inicio de cada segundo. De esta forma, el ordenador puede capturar la marca de tiempo de su reloj de sistema cuando llega este flanco, y etiquetarla con la hora del GPS cuando llegue el mensaje GPRMC (o similar).

Existen en el mercado receptores GPS con interfaz USB, sin embargo, la señal PPS no está disponible de ninguna forma, por lo que no son recomendables para este caso de uso.

3.4. Software para comunicación con receptores GPS

ntpd ya incluye un driver NMEA, por lo que no es necesario software adicional. En el caso de Linux, además tenemos la opción de usar el servicio `gpsd`, que además de NMEA también soporta UBX. De hecho, `chrony` no dispone de drivers sino que se integra con `gpsd` (a través de un *socket* local) para el caso en que queremos usar relojes GPS como referencia.

La ventaja en este caso es que `gpsd` también gestiona la geolocalización, de forma que cualquier aplicación puede consultarla a través de este servicio en lugar de comunicarse directamente con el receptor GPS. Esto no es posible tan fácilmente en el caso de `ntpd`: al tener capturado el puerto de comunicación con el receptor GPS, la única forma de acceder a la geolocalización es usar el comando `ntpq -c clockvar`, cuya salida incluye el último mensaje NMEA recibido.

4. Bases de tiempo para ocultaciones

Actualmente, todas las soluciones para implementar una base de tiempos de precisión suficiente para la observaciones de ocultaciones usan señales GPS. Se pueden resumir en las siguientes [3]:

- Cámara CMOS con obturador global y GPS integrado. Probablemente la solución idónea: los *frames* capturados directamente se etiquetan con las marcas de tiempo obtenidas por GPS, todo ello en la propia cámara.
- Insertador de tiempo en vídeo. Usado con cámaras de salida de vídeo analógica para sobreimpresionar la marca de tiempo, obtenido mediante GPS, sobre cada *frame* de vídeo, antes de ser procesado por la tarjeta capturadora para digitalizarlo.
- Servidor dedicado NTP basado en GPS. Muy extendida en estos últimos años, consiste en usar un SBC (*Single Board Computer*), p.ej. Raspberry Pi, al que se conecta un receptor GPS con UART y PPS, y en el que se despliega un servicio NTP². Es sencilla de implementar y de bajo coste, y obtiene una precisión por debajo del milisegundo [12]. El equipo de captura debe configurarse para usar este servidor como referencia principal, y ambos deben estar conectados preferiblemente por ethernet.
- *Flasher*. Dispositivo que emite destellos regulares antes y después del evento a observar, en instantes conocidos. El dispositivo se instala físicamente de manera que los destellos se registran en la cámara que captura el evento, y son analizados posteriormente mediante software para asociar a cada *frame* su marca de tiempo, interpolada a partir de aquellas en las que está presente el destello. Dentro de esta categoría existe una aplicación para Android, denominada *Occult Flash Tag*, que emite los destellos con el LED del *smartphone* (aunque usa NTP para la sincronización horaria).

5. Descripción de la solución propuesta

La solución que aquí se propone, que no pretende sustituir a las ya mencionadas, es la de conectar un módulo receptor GPS de bajo coste, con interfaz UART, usando un conversor a USB y transportando la señal PPS a través de una de las líneas de control. No se trata de una propuesta original, ya que ha sido implementada en unos pocos productos comerciales, pero que actualmente resultan muy difíciles de adquirir. Esta solución se ha descartado de forma recurrente en parte por este motivo, pero sobre todo, por las dudas que surgen al considerar que el transporte USB pueda afectar negativamente a su precisión.

La aportación del presente trabajo es compilar ordenadamente y de forma contrastada todos los pasos necesarios para su implementación, y especialmente, mostrar los resultados de las pruebas realizadas para determinar la precisión alcanzable.

Los siguientes apartados detallan el **montaje de hardware** realizado y la **configuración base** para probar el correcto funcionamiento del dispositivo y del software, tanto en Linux como en Windows. Los pasos necesarios para la **calibración de la fuente GPS** son los siguientes:

1. Medir el desfase instrumental que presenta el fuente de tiempo GPS desde la perspectiva del servicio NTP. Este desfase se debe principalmente a las latencias de todas las capas de software intermedias. Para realizar esta medición usaremos múltiples fuentes NTP stratum 1 para construir una referencia lo más fidedigna posible. En este sentido es importante que la conexión a Internet que usemos sea lo más estable posible, por lo que es recomendable disponer de una línea de fibra óptica y conectar el equipo de captura mediante ethernet al router, evitando usar Wi-Fi.

²Raspberry Pi 5 añade soporte hardware para marcaje de tiempo, lo que permite desplegar un servicio PTP. No obstante, la NIC no dispone de entrada PPS dedicada por lo que la señal PPS proveniente del receptor GPS debe ir conectada a un pin de entrada/salida de propósito general (GPIO) y ser gestionada por software. A pesar de ello, experimentos recientes [13] sugieren que los resultados pueden ser mejores que con NTP. En cualquier caso, las NIC de equipos portátiles no soportan marcaje de tiempo por hardware y el cliente PTP de Windows y su interacción con los drivers todavía necesita madurar [15].

2. Una vez obtenida una estimación de dicho desfase, aplicaremos una corrección del mismo en la configuración del servicio NTP. Gracias a ello, sólo quedará una fluctuación residual o *jitter*, que nos indicará la precisión alcanzable.

La estimación del desfase se realiza con un script que analiza los logs de estadísticas generados por el servicio NTP. Se encuentra disponible en el repositorio asociado al artículo [1].

5.1. Montaje hardware

La conexión con el ordenador se ha realizado con un módulo USB-serie del fabricante FTDI, basado en el chip FT232RL³. Se trata de un módulo de precio muy reducido y fácilmente localizable en el mercado. Por otra parte, este módulo está plenamente soportado en Linux y Windows (aunque en este último caso requiere instalar manualmente los drivers VCP disponibles en la web del fabricante⁴). Sin embargo, el motivo principal para seleccionar este fabricante es que, según la documentación oficial [17], la latencia del driver es mínima cuando se detecta un cambio en las líneas de control. Esto sugiere que quizá sea posible transportar la señal PPS a través de USB sin añadir excesiva incertidumbre⁵.

En particular, conectaremos PPS con la línea de control DCD (*Data Carrier Detect*), ya que es la habitualmente usada con *gpsd* y el driver NMEA de *ntpd*. Para la recepción de mensajes conectaremos la línea TX del módulo BE-280 con RX de FT232RL, e incluiremos también la conexión recíproca (RX BE-280 con TX FT232RL) para tener la posibilidad de controlar el módulo GPS. Por último, y dado que el consumo de éste es muy bajo, podemos alimentarlo directamente desde el módulo FT232RL, asegurándonos que el selector de tensión (en caso de existir) esté en la posición 5V. El conexionado físico, que requiere soldar varios cables, se muestra en la figura 3.

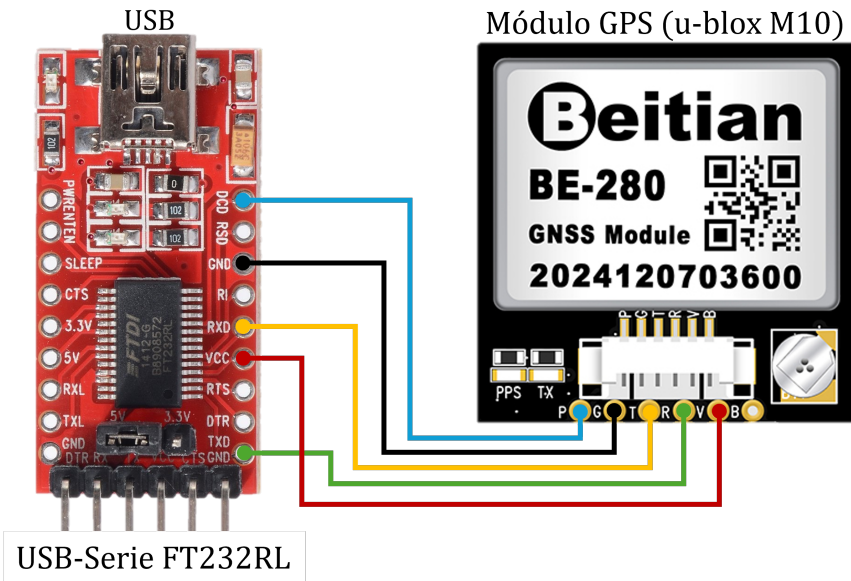


Figura 3. Esquema de conexionado.

³La medición de desfase se ha probado también con un módulo basado en el chip FT232H, que implementa USB 2.0 Hi-Speed, si bien los resultados han sido similares

⁴<https://ftdichip.com/drivers/vcp-drivers/>

⁵Existe cierto debate respecto a la conveniencia de cambiar la configuración del *latency timer* del driver FTDI, reduciendo su valor (por defecto, 16 ms) al mínimo (1 ms). Este temporizador está ligado al buffer de recepción, y sirve para poner un límite a la espera de la llegada de bytes a través del puerto serie. Dado que no se usa en el caso de las líneas de control, no tiene efecto en la solución propuesta.

Una cuestión a tener en cuenta es que USB 3 produce emisiones de radio en el rango de frecuencias que se usan para la recepción de señales GPS [18]. Por ello es necesario alejar en lo posible el módulo, que lleva la antena integrada, del equipo de captura y de la cámara.

Para proteger el conjunto es conveniente preparar una carcasa apta para intemperie, teniendo en cuenta que la antena debe quedar orientada hacia el cielo. La figura 4 muestra una carcasa impresa en 3D atada a una pata del telescopio, alejando el receptor de posibles interferencias. La parte que queda horizontal contiene el módulo GPS con la antena orientada hacia arriba. El diseño está disponible en el repositorio [1].



Figura 4. Dispositivo alojado en carcasa impresa en 3D.

5.2. Implementación en Linux

Las instrucciones de instalación y configuración que se desarrollan en esta sección han sido probadas en una instalación de Ubuntu 22.04 LTS y posteriormente validadas sobre Ubuntu 24.04 LTS.

5.2.1. Configuración base con `gpsd` y `chrony`

En primer lugar prepararemos la configuración de `gpsd` [19] y validaremos que se comunica correctamente con el hardware. Instalamos los paquetes de software necesarios y preparamos el usuario de sistema para el servicio:

```
$ sudo apt install gpsd gpsd-clients pps-tools
$ sudo dpkg-reconfigure gpsd
Creating/updating gpsd user account...
gpsd.socket is a disabled or a static unit not running, not starting it.
gpsd.service is a disabled or a static unit not running, not starting it.
gpsd.socket is a disabled or a static unit not running, not starting it.
```

Al conectar el módulo GPS, podemos comprobar a qué nodo de dispositivo se le asocia en el sistema:

```
$ sudo dmesg
[...]  
... usb 1-5: FTDI USB Serial Device converter now attached to ttyUSB0  
[...]
```

En este caso es `/dev/ttyUSB0`. En este punto podemos comprobar si `gpsd` reconoce y se comunica con el dispositivo, lanzando el servicio manualmente en modo de depuración:

```
$ sudo gpsd -D 5 -N -n /dev/ttyUSB0  
gpsd:INFO: launching (Version 3.25, revision 3.25)  
gpsd:INFO: starting uid 0, gid 0  
gpsd:INFO: Command line: gpsd -D 4 -N -n /dev/ttyUSB0  
gpsd:INFO: listening on port gpsd  
gpsd:PROG: NTP:SHM: shmctl(32769,0,0) succeeded, unit 0  
gpsd:PROG: NTP:SHM: shmctl(32770,0,0) succeeded, unit 1  
gpsd:PROG: NTP:SHM: shmctl(32772,0,0) succeeded, unit 2  
gpsd:PROG: NTP:SHM: shmctl(32774,0,0) succeeded, unit 3  
gpsd:PROG: NTP:SHM: shmctl(32775,0,0) succeeded, unit 4  
gpsd:PROG: NTP:SHM: shmctl(32776,0,0) succeeded, unit 5  
gpsd:PROG: NTP:SHM: shmctl(32777,0,0) succeeded, unit 6  
gpsd:PROG: NTP:SHM: shmctl(32779,0,0) succeeded, unit 7  
gpsd:PROG: NTP:SHM: shmctl(32780,0,0) succeeded, unit 8  
gpsd:PROG: NTP:SHM: shmctl(32781,0,0) succeeded, unit 9  
gpsd:PROG: NTP:SHM: shmctl(32782,0,0) succeeded, unit 10  
gpsd:PROG: NTP:SHM: shmctl(32784,0,0) succeeded, unit 11  
gpsd:PROG: successfully connected to the DBUS system bus  
gpsd:PROG: SHM: shmget(0x47505344, 29240, 0666) for SHM export succeeded  
gpsd:PROG: SHM: shmctl() for SHM export succeeded, segment 32785  
gpsd:INFO: stashing device /dev/ttyUSB0 at slot 0  
gpsd:PROG: CORE: open_device(/dev/ttyUSB0) fd -1  
gpsd:PROG: CORE: gpsd_activate(/dev/ttyUSB0, 2) fd -1  
gpsd:PROG: CORE: no /etc/gpsd/device-hook present, skipped running ACTIVATE hook.  
No such file or directory(2)  
gpsd:PROG: CORE: gpsd_open(/dev/ttyUSB0) fd -1  
gpsd:PROG: SER: gpsd_serial_open(/dev/ttyUSB0) sourcetype 3 fd -1  
gpsd:INFO: SER: opening GPS data source type 3 at '/dev/ttyUSB0'  
gpsd:INFO: SER: fd 7 current speed 38400, 8N1  
gpsd:PROG: CORE: Probing "Garmin USB binary" driver...  
[...]  
gpsd:PROG: CORE: selecting u-blox driver...  
gpsd:INFO: CORE: /dev/ttyUSB0 identified as type u-blox, 1 sec  
gpsd:INFO: CORE: /dev/ttyUSB0 38400bps  
[...]
```

Este comando genera mucha información por lo que debemos cancelarlo tras pocos segundos con `Ctrl+C` para poder analizar la salida. Tras la inicialización del servicio, `gpsd` trata de determinar el tipo de receptor GPS y la velocidad del puerto serie. En este caso determina que se trata de un módulo u-blox (`gpsd:INFO: CORE: /dev/ttyUSB0 identified as type u-blox`) que está enviando mensajes a 38400 bps (`gpsd:INFO: CORE: /dev/ttyUSB0 38400bps`).

Es importante que confirmar en este punto que se está usando *kernel PPS* (KPPS), es decir, que el seguimiento de la señal PPS se está realizando mediante un módulo de *kernel* del sistema operativo, lo que proporciona, en condiciones normales, menor latencia y *jitter*. Para ello debemos localizar, en la salida del comando anterior, el siguiente mensaje:

```
[...]  
gpsd:INFO: KPPS:/dev/ttyUSB0 kernel PPS will be used  
[...]
```

En caso contrario, podemos investigar si la señal PPS está llegando por otra línea de control y por ello no es detectada por `gpsd`. En este caso, `ppsccheck` avisa de que `/dev/ttyUSB0` no es un dispositivo KPPS, ya que se trata de un puerto serie virtual, y crea el dispositivo KPPS `/dev/pps0` asociado a éste. En la salida del comando debemos ver que aparece, cada segundo, los eventos de `clear` y `assert`, es decir, los cambios de nivel de la señal PPS, a través de la línea `DCD`, `TIOCM_CD`.

```
$ sudo ppscheck /dev/ttyUSB0
WARNING: time_pps_create(/dev/ttyUSB0)) failed: Operation not supported(95)
WARNING: /dev/ttyUSB0 does not appear to be a KPPS device
INFO: matching /dev/pps0 opened

# Src      Seconds      Signals

KPPS 0.000000000 assert 0
KPPS 1759603810.001042068 clear 1
TTY 1759603810.001276456
KPPS 1759603810.101000282 assert 1
TTY 1759603810.101121817 TIOCM_CD

KPPS 1759603811.001014980 clear 2
TTY 1759603811.001167710
KPPS 1759603811.100972784 assert 2
TTY 1759603811.101108597 TIOCM_CD
[...]
```

A continuación debemos crear el fichero de configuración para `gpsd` a partir de la información obtenida, `/etc/default/gpsd`:

```
# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB0"

# Other options you want to pass to gpsd
GPSD_OPTIONS="-n -s38400"

# Automatically hot add/remove USB GPS devices via gpsdctl
USB AUTO="false"
```

Seguidamente arrancamos el servicio:

```
$ sudo systemctl start gpsd
```

Y comprobamos que está recibiendo correctamente la solución de posición, velocidad y hora, véase la figura 5.

```
$ cgps -s -u m # para salir, pulsar q
```

`gpsd` debe haber podido crear el dispositivo PPS, `/dev/pps0`. Podemos comprobarlo con `ppstest`:

```
$ sudo ppstest /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1759606048.101202976, sequence: 9 - clear 1759606049.001205442,
sequence: 10
source 0 - assert 1759606049.101212416, sequence: 10 - clear 1759606049.001205442,
sequence: 10
source 0 - assert 1759606049.101212416, sequence: 10 - clear 1759606050.001215413,
sequence: 11
[...]
```

A continuación instalamos `chrony` y generamos los ficheros de configuración:

					Seen 34/Used 14						
Time	2025-10-04T19:53:09.000Z (18)				GNSS	S	PRN	Elev	Azim	SNR	Use
Latitude	. N				GP	25	25	54.0	52.0	17.0	Y
Longitude	. W				GP	29	29	81.0	148.0	37.0	Y
Alt (HAE, MSL)	79.395,	32.218	m		GP	32	32	38.0	233.0	39.0	Y
Speed	0.13		km/h		GA	2	302	36.0	60.0	15.0	Y
Track (true, var)	356.6,	-1.1	deg		GA	7	307	49.0	255.0	32.0	Y
Climb	5.94		m/min		GA	8	308	28.0	191.0	35.0	Y
Status	3D DGPS FIX (9 secs)				GA	27	327	28.0	274.0	17.0	Y
Long Err (XDOP, EPX)	0.48,	+/-	1.7	m	GA	30	330	68.0	333.0	18.0	Y
Lat Err (YDOP, EPY)	0.67,	+/-	2.4	m	GA	34	334	25.0	123.0	37.0	Y
Alt Err (VDOP, EPV)	1.24,	+/-	2.7	m	BD	19	419	12.0	234.0	27.0	Y
2D Err (HDOP, CEP)	0.82,	+/-	2.1	m	BD	27	427	54.0	84.0	29.0	Y
3D Err (PDOP, SEP)	1.49,	+/-	6.6	m	BD	28	428	22.0	45.0	16.0	Y
Time Err (TDOP)	0.83				BD	30	430	43.0	175.0	36.0	Y
Geo Err (GDOP)	1.70				BD	37	437	21.0	102.0	21.0	Y
Speed Err (EPS)		+/-	18.5	km/h	GP	11	11	15.0	49.0	14.0	N
Track Err (EPD)	n/a				GP	12	12	26.0	69.0	0.0	N
Time offset	0.373328026		s		GP	18	18	8.0	169.0	24.0	N
Grid Square	IM77bj01				GP	24	24	7.0	136.0	18.0	N
ECEF X, VX	5047391.020	m	0.060	m/s	GP	26	26	15.0	280.0	0.0	N
ECEF Y, VY	-522928.650	m	-0.010	m/s	GP	28	28	57.0	314.0	19.0	N
ECEF Z, VZ	3851243.240	m	0.000	m/s	GP	31	31	26.0	313.0	0.0	N
					SB123	36	32.0	128.0	0.0	0.0	N
					SB126	39	7.0	103.0	0.0	0.0	N
					SB127	40	15.0	108.0	31.0	0.0	N
					SB136	49	45.0	162.0	45.0	0.0	N
					GA	11	311	1.0	25.0	0.0	N
					GA	14	314	15.0	50.0	13.0	uN
					GA	15	315	3.0	168.0	0.0	N
					GA	29	329	21.0	314.0	0.0	N
					GA	36	336	23.0	67.0	0.0	N
					BD	5	405	10.0	108.0	28.0	N
					BD	22	422	15.0	285.0	0.0	N
					More...						

Figura 5. Información mostrada por el comando *cgps*.

```
$ sudo apt install chrony

$ echo "server hora.roa.es iburst prefer" | sudo tee
/etc/chrony/sources.d/st1.sources
$ echo "refclock SOCK /var/run/chrony.ttyUSB0.sock refid GPS" | sudo tee
/etc/chrony/conf.d/01-gpsd.conf
$ echo "log tracking statistics measurements" | sudo tee
/etc/chrony/conf.d/02-log.conf
$ echo "logbanner 0" | sudo tee -a /etc/chrony/conf.d/02-log.conf
```

El fichero *st1.sources* contendrá las fuentes NTP, por ahora solo el servidor de ROA (Real Observatorio de la Armada), *hora.roa.es*. En *01-gpsd.conf* estamos indicando que la interacción con *gpsd* se realizará mediante el socket */var/run/chrony.ttyUSB0.sock*, y que el reloj de referencia asociado tendrá como identificador GPS. En *02-log.conf* indicamos que se deben generar distintos tipos de logs, entre ellos el de estadísticas, que nos permitirá estudiar el desfase de la fuente GPS; la directiva *logbanner 0* desactiva la generación repetitiva de los rótulos de las columnas de datos en los ficheros de log, que entorpecería su análisis automatizado.

Reiniciamos los servicios, primero *chrony* para que lea los ficheros de configuración y genere el socket, y a continuación *gpsd* para que lo detecte y haga uso del mismo.

```
$ sudo systemctl daemon-reload # necesario tras modificar ficheros de
    configuracion de chrony
$ sudo systemctl restart chronyd
$ sudo systemctl restart gpsd
```

Al iniciar chrony, éste registra si ha forzado un ajuste "de un salto" al reloj del sistema, p. ej.:

```
$ sudo systemctl status chrony
[...]
...: System clock wrong by -2.308569 seconds
...: System clock was stepped by -2.308569 seconds
```

Para comprobar el estado de chrony podemos usar el cliente chronyc [20], añadiendo los argumentos tracking, sources y/o sourcestats para obtener información detallada sobre la fuente seleccionada como primaria, el resumen de las fuentes y estadísticas sobre las mismas. El parámetro -v muestra la leyenda de las columnas para estas dos últimas opciones.

```
$ chronyc -n -m tracking 'sources -v' 'sourcestats -v'
Reference ID : 47505300 (GPS)
Stratum      : 1
Ref time (UTC) : Sat Oct 04 20:18:34 2025
System time   : 0.000000075 seconds fast of NTP time
Last offset   : -0.000000140 seconds
RMS offset    : 0.000000281 seconds
Frequency     : 18.246 ppm slow
Residual freq : -0.000 ppm
Skew          : 0.005 ppm
Root delay    : 0.000000001 seconds
Root dispersion : 0.000016294 seconds
Update interval : 16.0 seconds
Leap status   : Normal

.-- Source mode '^' = server, '=' = peer, '#' = local clock.
/.- Source state '*' = current best, '+' = combined, '-' = not combined,
|/ 'x' = may be in error, '~' = too variable, '?' = unusable.
||
||      .- xxxx [ yyyy ] +/- zzzz
||      |xxxx = adjusted offset,
||      |yyyy = measured offset,
||      |zzzz = estimated error.
||      \
||      |
||      |
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
#* GPS                  0  4  377  13 -883ns[-1021ns] +/- 1638ns
^~ 150.214.94.5         1  7  377  97 +382us[+382us] +/- 17ms

      .- Number of sample points in measurement set.
      /.- Number of residual runs with same sign.
      | /.- Length of measurement set (time).
      | /.- Est. clock freq error (ppm).
      | /.- Est. error in freq.
      | /.- Est. offset.
      | | On the -.
      | | samples. \
      | |
Name/IP Address      NP  NR  Span Frequency Freq Skew Offset Std Dev
=====
GPS                  46  21  721  -0.000    0.005  -0ns 2456ns
150.214.94.5         9   4 1032  -0.074    0.571 +640us 116us
```

Vemos que se ha seleccionado la fuente GPS, de tipo local clock, como principal (current best), mientras que el servidor de ROA se queda fuera del algoritmo de combinación (not combined), ya que comparativamente presenta peores estadísticas de desfase (offset) e incertidumbre. Destaca el bajo valor de incertidumbre (Std Dev, *Standard Deviation*) obtenido, de tan sólo 2456 ns.

No es necesario configurar el socket `chrony.ttyUSB0.sock` explícitamente en `gpsd`, pero para que funcione la comunicación con `chrony` es necesario asegurarse de que éste se ha iniciado antes que `gpsd`, tal como hemos hecho anteriormente de forma manual. El orden de arranque de los servicios se establece como una regla `After` en `gpsd`, que definimos en el fichero `/etc/systemd/system/gpsd.service.d/gpsd-after-chronyd.conf`:

```
[Unit]
After=chronyd.service
```

Finalmente, solo quedaría habilitar el arranque automático de estos servicios:

```
$ sudo systemctl enable chrony
$ sudo systemctl enable gpsd
```

5.2.2. Medición del desfase de la fuente GPS PPS

Modificamos la definición del reloj de referencia GPS para que únicamente sea monitorizado, sin que `chrony` pueda seleccionarlo, editando `/etc/chrony/conf.d/01-gpsd.conf`:

```
refclock SOCK /var/run/chrony.ttyUSB0.sock refid GPS noselect
```

Añadimos un conjunto de servidores NTP stratum 1 que conjuntamente servirán para contruir la hora NTP de referencia sobre la cual medir el desfase de la fuente GPS. En particular, para la prueba realizada, `/etc/chrony/sources.d/st1.sources` se configuró con el siguiente contenido:

```
server hora.roa.es iburst maxpoll 7
server minuto.roa.es iburst maxpoll 7
server hora.cica.es iburst maxpoll 7
server ntps1-0.cs.tu-berlin.de iburst maxpoll 7
server rustime01.rus.uni-stuttgart.de iburst maxpoll 7
server rustime02.rus.uni-stuttgart.de iburst maxpoll 7
server ntp.vsl.nl iburst maxpoll 7
server ntp.fizyka.umk.pl iburst maxpoll 7
server tempus1.gum.gov.pl iburst maxpoll 7
server tempus2.gum.gov.pl iburst maxpoll 7
server time1.stupi.se iburst maxpoll 7
server ntp1.inrim.it iburst maxpoll 7
server ntp1.oma.be iburst maxpoll 7
server time.esa.int iburst maxpoll 7
server time1.google.com iburst maxpoll 7
server time2.google.com iburst maxpoll 7
server time.windows.com iburst maxpoll 7
server time1.apple.com iburst maxpoll 7
server time2.apple.com iburst maxpoll 7
server gbg1.ntp.se iburst maxpoll 7
server mm01.ntp.se iburst maxpoll 7
server sth1.ntp.se iburst maxpoll 7
server ntp.time.nl iburst maxpoll 7
```

Para que tengan efecto los cambios:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart chronyd
```

El conjunto de servidores NTP elegidos puede tener bastante repercusión en la medición. En general es conveniente seleccionar servidores que tengan conectividad de latencia baja y a ser posible, estable (propia de proveedores de servicio). Por otra parte, debe monitorizarse el número de servidores seleccionados por `chrony`, mediante el comando `chronyc -n sources -v`: si se seleccionase un número

bajo de servidores, la medición podría quedar sesgada. Las fuentes seleccionadas aparecen con el símbolo '+', incluyendo también la principal, '**'.

Los ficheros de log de estadísticas generados por chrony, /var/log/chrony/statistics.log*, cuyo formato se describe con detalle en [21], sirven de base para el análisis de offset de la fuente GPS. Es conveniente acumular datos durante varios días si es posible para que las estadísticas tengan el menor sesgo posible.

Este análisis se ha implementado con un script Python [1] que obtiene el valor medio de desfase, $\bar{\theta}$, y su desviación estándar, σ , como medida de su dispersión o fluctuación. También se aplica *sigma clipping* (con factor 2.5 sobre la desviación estándar) para que los resultados sean robustos ante la presencia de *outliers*, obteniendo los estadísticos θ_{CLIP} y σ_{CLIP} .

La gráfica de la figura 6 muestra la evolución del desfase durante el intervalo de análisis (más de 8 días) y las estadísticas resultantes. El dato que nos interesa en este paso es el valor de **0.000855 s** para el desfase, que corregiremos en el apartado siguiente. Es notable que su desviación estándar es de tan sólo 94 μ s (70 μ s en el caso de *sigma clipping*), lo que nos indica que el valor de desfase obtenido tiene poca incertidumbre.

5.2.3. Corrección del desfase de la fuente GPS PPS

Eliminamos `noselect` y añadimos el valor de desfase obtenido durante el análisis al argumento `offset`, así como la opción `prefer` en /etc/chrony/conf.d/01-gpsd.conf:

```
refclock SOCK /var/run/chrony.ttyUSB0.sock refid GPS offset 0.000855 prefer
```

Aplicamos los cambios:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart chronyd
```

Dado que el tiempo de parada es muy breve, chrony tarda pocos minutos en volver a converger. La salida típica obtenida al consultar el estado las fuentes muestra que GPS es la fuente principal, con *jitter* de aproximadamente 1.5 μ s, inferior al de las fuentes NTP en varios órdenes de magnitud, por lo que el algoritmo de combinación no las considera:

```
$ chronyc -n sources
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
#* GPS                  0 4 377 18 +187ns[ +443ns] +/- 1468ns
^~ 194.58.203.20         1 6 377 37 -141us[ -140us] +/- 31ms
^~ 150.214.5.121         1 6 377 36 +43us[ +44us] +/- 15ms
^~ 150.214.94.5          1 6 377 42 +705us[ +706us] +/- 17ms
^~ 150.214.94.10         1 6 377 36 +752us[ +752us] +/- 17ms
^~ 194.58.204.20         1 6 377 37 -210us[ -210us] +/- 29ms
^~ 158.75.5.245          1 6 377 40 +90us[ +91us] +/- 30ms
^~ 94.198.159.10         1 6 377 35 +1550us[+1550us] +/- 21ms
^~ 31.223.173.226        1 6 377 40 +458us[ +459us] +/- 18ms
^~ 193.204.114.232       1 6 377 38 -3080us[-3079us] +/- 20ms
^~ 193.190.230.105       2 6 377 39 -2457us[-2456us] +/- 48ms
^~ 130.149.17.21         1 6 346 168 +2117us[+2117us] +/- 30ms
^~ 129.69.253.1          1 6 377 41 +1371us[+1372us] +/- 22ms
^~ 129.69.253.17         1 6 377 34 +631us[ +632us] +/- 23ms
^~ 194.58.202.20         1 6 377 36 -1017us[-1016us] +/- 30ms
^~ 194.146.251.100       1 6 377 37 -1756us[-1756us] +/- 31ms
^~ 194.146.251.101       1 6 377 35 -1182us[-1181us] +/- 32ms
^~ 192.171.1.150         1 6 377 35 +1304us[+1305us] +/- 21ms
^~ 51.145.123.29         3 6 377 36 +1855us[+1856us] +/- 34ms
^~ 216.239.35.0          1 6 377 36 +1240us[+1241us] +/- 15ms
^~ 192.36.143.150       1 6 377 35 +1118us[+1119us] +/- 31ms
^~ 216.239.35.4         1 6 377 37 +461us[ +462us] +/- 14ms
```

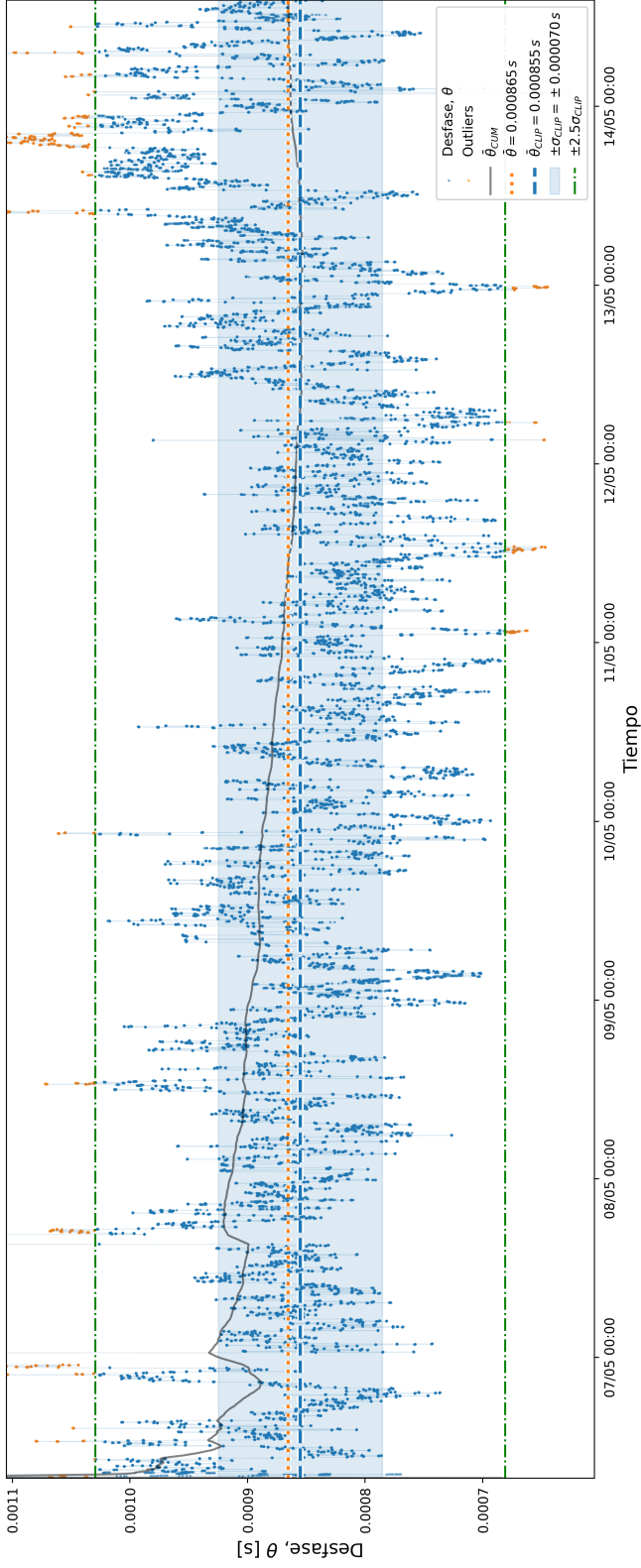


Figura 6. Evolución del desfase de la fuente GPS durante más de 8 días, medido en base a 21 fuentes NTP stratum 1. Prueba realizada en entorno Linux, con `gpsd` y `chrony`.

En cualquier caso, para comprobar si el desfase queda debidamente corregido y obtener una medición fiable del error, debemos dejar transcurrir varias horas antes de realizar un nuevo análisis con el script. Durante este lapso, se puede realizar la prueba de captura descrita para establecer si tiene algún impacto en el *jitter*.

5.3. Implementación en Windows

Para poder usar la señal PPS en Windows es necesario recurrir a la implementación de NTP realizada por el fabricante Meinberg. El software se descarga desde la sección 'NTP for current Windows versions' de su portal web [22]. La instalación es guiada y basta con dejar todas las opciones por defecto, salvo en el paso de creación del fichero de configuración, donde seleccionaremos 'Spain' en el campo 'Want to use predefined public NTP servers', para poder hacer una prueba inicial. El instalador finaliza arrancando el servicio NTP.

Para monitorizar de forma cómoda el servicio, podemos usar la herramienta 'NTP Time Server Monitor' (figura 7), también de Meinberg, disponible en su portal web [23]. Una vez instalada, es necesario ejecutarla *como administrador* si queremos tener control para parar o reiniciar el servicio, cambiar la configuración, acceder al log de eventos, etc.

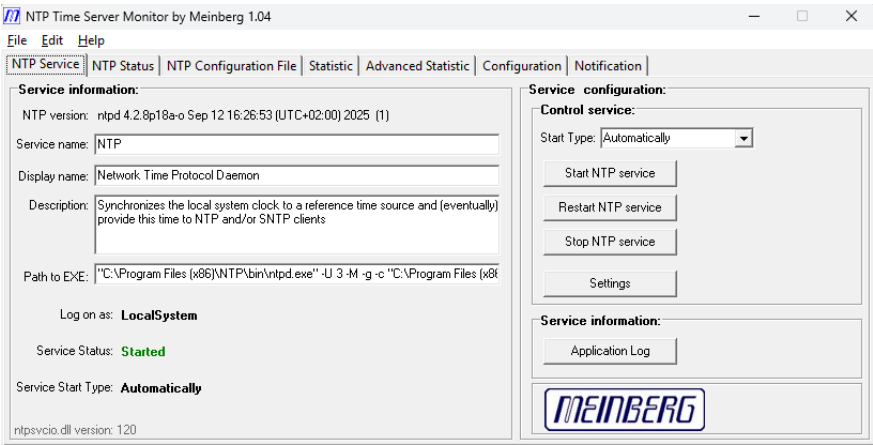


Figura 7. NTP Time Server Monitor.

5.3.1. Configuración en Windows con Meinberg NTP

Antes de configurar la fuente GPS debemos habilitar el *driver* PPS [24]. Para ello, debemos ejecutar la herramienta 'Editor de registro' de Windows (regedit.exe). En ella, navegamos hasta la entrada:

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTP

y creamos una entrada de tipo 'cadena múltiple' (REG_MULTI_SZ) con clave PPSProviders y valor loopback-ppsapi-provider.dll (guiones medios, no bajos). Esta nueva configuración requiere reiniciar el servicio NTP para que tenga efecto, cosa que podemos hacer desde 'NTP Time Server Monitor', pestaña 'NTP Service', botón 'Restart NTP Service'. En esta misma pestaña, pulsamos 'Application Log', que creará una nueva pestaña 'NTP Event Log'. En ella debe aparecer el evento:

```
time_pps_create: loaded
'C:\Program Files (x86)\NTP\bin\loopback-ppsapi-provider.dll'
```

que nos confirma que el *driver* PPS se ha cargado correctamente. Este *driver* se ejecuta en espacio de usuario, con menor nivel de privilegio y prioridad que los procesos de kernel del sistema operativo, lo cual puede afectar en el *jitter* observado [24].

Para configurar la fuente GPS necesitamos saber en primer lugar qué identificador tiene el dispositivo serie asociado y a qué velocidad está configurado. El primer dato se puede obtener del 'Administrador de dispositivos', desplegando la sección 'Puertos (COM & LPT)', donde debe aparecer alguna entrada 'USB Serial Port', indicando el identificador de puerto entre paréntesis, p.ej. 'USB Serial Port (COM12)'. Para conocer la velocidad de transmisión, tendremos que recurrir a la información del fabricante del módulo. En el caso de módulos u-blox como el usado para las pruebas, otra opción es recurrir al software 'u-center 2' [25]: en la sección 'Data sources', podemos configurar la conexión con el módulo pulsando el botón '+' y seleccionando el puerto (COM12 en este caso) y dejando la opción 'Enable autobauding'. Al conectar, podremos ver la velocidad con la que ha conseguido comunicarse con el módulo, figura 8.

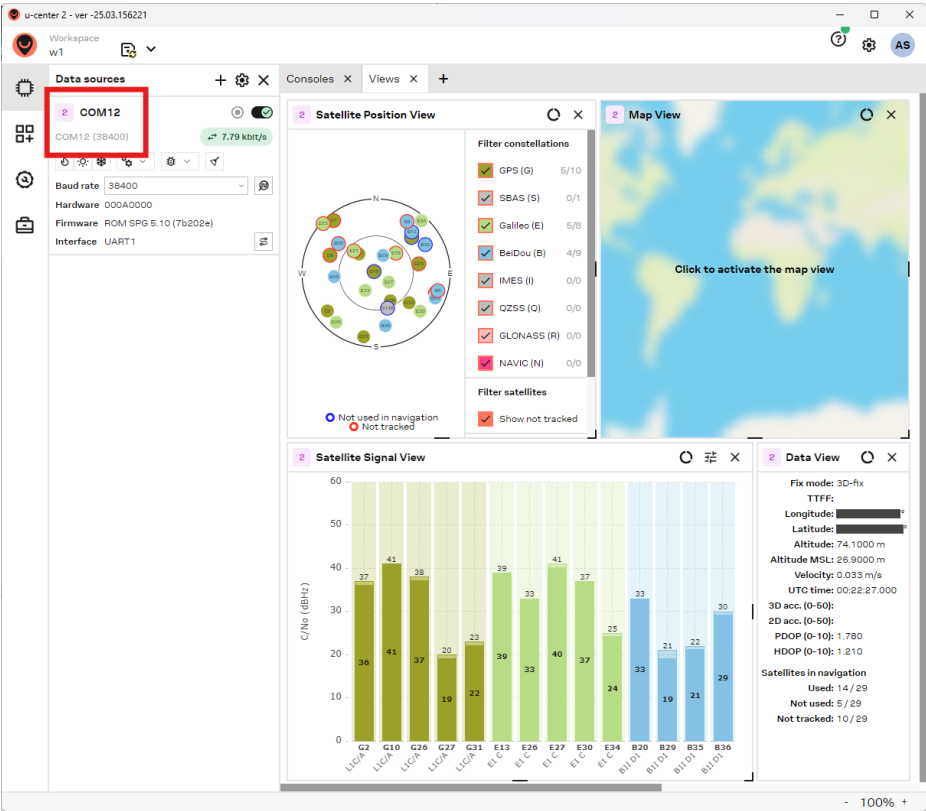


Figura 8. Uso de u-center 2 para obtener el puerto y velocidad del receptor GPS (solo para dispositivos u-blox).

A partir de esta información, puerto y velocidad, vamos a contruir los valores, un tanto crípticos, que deberán tener los parámetros de configuración del reloj basado en GPS. En primer lugar, en `ntpd`, el reloj GPS se declara con la directiva `server`, como si se tratase de una fuente NTP, pero usando una dirección IP ficticia basada en el número de puerto: `127.127.20.x`, donde el octeto `20` especifica el *driver* NMEA [26], y `x` corresponde al número de puerto, es decir, 12 para COM12. El siguiente parámetro a indicar es `mode`, que se construye sumando los códigos correspondientes a las opciones que

queremos habilitar: en nuestro caso, 49, como suma de 1, que indica que se procesarán los mensajes NMEA de tipo GPRMC (*Recommended Minimum Data*), y de 48, que indica que la velocidad es de 38400 bps. Los valores disponibles están documentados en la referencia del driver NMEA [26]. Usaremos un sondeo de 16 ($= 2^4$) segundos, que indicaremos con el valor 4 en los parámetros `minpoll` y `maxpoll`. Para especificar que debe asociarse a esta fuente una señal PPS hay que añadir una directiva `fudge` asociada a la misma dirección IP ficticia, en la que activaremos el parámetro `flag1` (*enable PPS signal processing*) y que más adelante nos permitirá compensar el desfase medido a través del parámetro `offset`. En definitiva, las líneas que debemos añadir al fichero de configuración ya existente, `C:\Program Files (x86)\NTP\etc\ntp.conf`, son las siguientes:

```
# GPS PPS via COM12
# mode 49 = 1 + 48 = process $GPRMC + linespeed 38400 bps
server 127.127.20.12 mode 49 minpoll 4 maxpoll 4
# flag1 = PPS, time1 = offset calibration
fudge 127.127.20.12 time1 0.0 flag1 1
```

El fichero puede editarse desde la pestaña 'NTP Configuration File' de la herramienta 'NTP Time Server Monitor', que al salvar nos dará opción a reiniciar el servicio para aplicar los cambios. Hay que recordar que para otro puerto y/o velocidad es necesario revisar la IP ficticia y el valor de `mode` según lo indicado.

Tras el reinicio del servicio, podemos recurrir de nuevo a la herramienta de monitorización, pestaña 'NTP Status', para observar el listado de fuentes y su stratum, frecuencia de sondeo, disponibilidad, etc. En este punto, nos interesa especialmente consultar el estado de la fuentes, en particular de la fuente GPS, que se codifica un carácter: 'o' nos indicaría que esta fuente está seleccionada y tiene asociada una señal PPS (figura 9), mientras que '*' nos indicaría que la señal PPS no se detecta. El significado de los códigos se puede consultar pulsando el botón 'Legend'.

NTP Time Server Monitor by Meinberg 1.04

File Edit Help

NTP Service NTP Status NTP Configuration File Statistic Advanced Statistic Configuration NTP Event Log Notification

Localhost

Current local NTP Status: Sync to: 212.227.232.46 Offset: -0.285ms Stratum: 1 Refresh Interval: 10 s

NTP Status:

Remote	Refid	Stratum	Type	When	Poll	Reach	Delay	Offset	Jitter
o 127.127.20.12	GPS	0	Local clock	11	16	377	0.000	-98.443	0.623
92.113.12.77	INIT	16	Unicast server	6165d	64	000	0.000	0.000	0.000
* 212.227.232.46	193.134.29.11	2	Unicast server	44	64	001	17.068	-0.285	1.290
+ 162.159.200.123	10.111.8.80	3	Unicast server	43	64	001	10.556	-0.563	1.288
+ 91.224.149.41	145.238.80.80	2	Unicast server	42	64	001	32.079	0.994	1.144
- 91.194.60.128	131.188.3.220	2	Unicast server	41	64	001	28.689	1.687	1.411

Polling Status: Running NTP Version: ntpd 4.2.8p18a-o Sep 12 16:26:53 (UTC+02:00) 2025 (1) DNS lookup Legend

Figura 9. Fuente GPS seleccionada, verificando que se detecta la señal PPS.

Es muy posible que la fuente GPS se seleccione por un momento, mostrando el indicador PPS 'o' (*PPS source of server is selected for synchronization*) y por tanto confirmando que el montaje hardware y la configuración es correcta, pero que unos minutos después quede marcada como *false ticker* (figura 10), 'x', ya que, al no haber corregido todavía su desfase, la hora que reporta estará en desacuerdo con el resto de fuentes. En cualquier caso, si tenemos dudas, podemos repetir la prueba eliminando todas las fuentes NTP del fichero de configuración (líneas `server` generadas durante la instalación de NTP), dejando solo la fuente GPS.

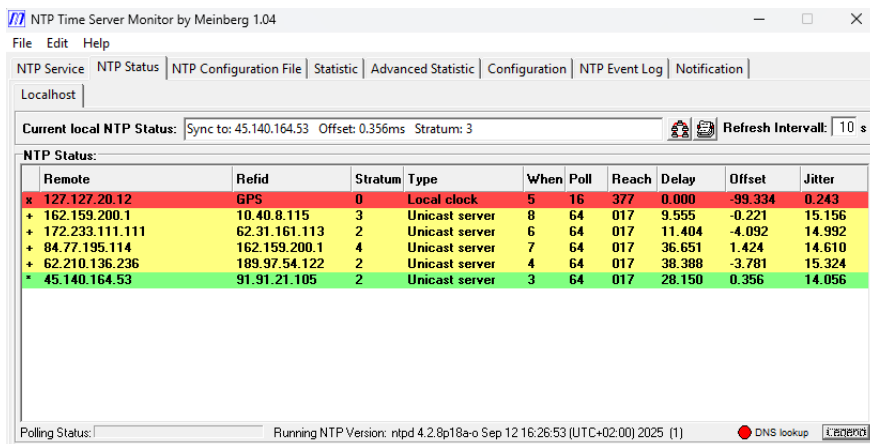


Figura 10. Fuente GPS marcada como falseticker.

También es posible consultar el estado de las fuentes con el comando `ntpq -n -p`.

5.3.2. Medición del desfase de la fuente GPS PPS

Al igual que con `chrony`, para la medición del desfase añadimos el parámetro `noselect` a la línea que define la fuente GPS:

```
server 127.127.20.12 mode 49 minpoll 4 maxpoll 4 noselect
```

Igualmente, añadimos un conjunto de servidores NTP stratum 1 para realizar la monitorización. En este caso, indicamos de forma explícita que el algoritmo de selección acepte hasta 20 servidores (`maxclock 20`) y nos aseguramos de habilitar los logs `peerstats` (entre otros), que nos servirán para analizar el desfase.

```
# Servidores stratum 1
server hora.roa.es iburst maxpoll 7
server minuto.roa.es iburst maxpoll 7
server hora.cica.es iburst maxpoll 7
server ntpsl-0.cs.tu-berlin.de iburst maxpoll 7
server rustime01.rus.uni-stuttgart.de iburst maxpoll 7
server rustime02.rus.uni-stuttgart.de iburst maxpoll 7
server ntp.vsl.nl iburst maxpoll 7
server ntp.fizyka.umk.pl iburst maxpoll 7
server tempus1.gum.gov.pl iburst maxpoll 7
server tempus2.gum.gov.pl iburst maxpoll 7
server time1.stupi.se iburst maxpoll 7
server ntp1.inrim.it iburst maxpoll 7
server ntp1.oma.be iburst maxpoll 7
server time.esa.int iburst maxpoll 7
server time1.google.com iburst maxpoll 7
server time2.google.com iburst maxpoll 7
server time.windows.com iburst maxpoll 7
server time1.apple.com iburst maxpoll 7
server time2.apple.com iburst maxpoll 7
server gbg1.ntp.se iburst maxpoll 7
server mm01.ntp.se iburst maxpoll 7
server sth1.ntp.se iburst maxpoll 7
server ntp.time.nl iburst maxpoll 7
```

```
# Permitir seleccion de hasta 20 servidores
tos maxclock 20
```

```
# Configuración de logs
enable stats
statsdir "C:\Program Files (x86)\NTP\etc\"
statistics peerstats loopstats
filegen peerstats file peerstats type day enable
```

Tras reiniciar el servicio, veremos que la fuente GPS queda descartada (sin código de estado), tal como hemos configurado. El servicio `ntpd` tiene una convergencia más lenta que `chrony` por lo que es posible que observemos valores de desfase altos o muy variables que pueden tardar horas en estabilizarse. Por lo tanto, al igual que con `chrony`, también es recomendable realizar la medición durante varios días.

En este caso se ha observado que la fuente principal cambia con frecuencia, y a consecuencia de ello, también cambia frecuentemente el conjunto de fuentes seleccionadas, a diferencia del comportamiento de `chrony`.

El análisis del desfase se realiza con un script similar al descrito para `chrony`, adaptándolo al formato de log usado por `ntpd`.

La gráfica de la figura 11 muestra la evolución del desfase durante el intervalo de análisis (más de 8 días) y las estadísticas resultantes. El dato que nos interesa en este paso es valor de **-0.100183 s** para el desfase, que corregiremos en el apartado siguiente.

La gráfica muestra mayor variabilidad en comparación con la obtenida en Linux: nótese la diferencia en la escala vertical de las gráficas, 0.5 ms para la de Linux frente a 5 ms para la de Windows. Aunque puede afectar el hecho de que el *driver* PPS trabaje en espacio de usuario, también hay que considerar que el continuo cambio de selección de fuentes puede haber contribuido a incrementar la desviación estándar observada, que alcanza el valor de 0.84 ms en el caso de *sigma clipping*. En este punto lo importante es que el valor de desfase obtenido sea representativo, y estos datos nos indican que debe serlo con bastante fiabilidad en un margen inferior a ± 1 ms. Una vez configurada la fuente GPS como principal, la selección será estable, y el *jitter* quedará afectado principalmente por las capas de transmisión vía USB y de software (*drivers* PPS y NMEA, y `ntpd`) y la posible carga de CPU del equipo.

Puede resultar sorprendente que los valores de desfase obtenidos sean tan diferentes para Linux y Windows, a pesar de usar el mismo hardware. Hay que tener en cuenta que no son comparables ya que en el primer caso, el reloj de referencia GPS que ve `chrony` se corresponde con el ofrecido por `gpsd`, que se encarga de toda comunicación con el receptor GPS y su sincronización con PPS, mientras que en el segundo caso, es `ntpd` quien se encarga de sincronizar PPS con los mensajes que le llegan desde el receptor GPS a través del driver NMEA.

5.3.3. Corrección del desfase de la fuente GPS PPS

La configuración final se obtiene realizando los siguientes cambios:

- sustituyendo `noselect` por `prefer`; esto de manera efectiva deshabilita el algoritmo de combinación de NTP, de tal forma que los valores de desfase y su error asociado serán los de la fuente GPS, que sistemáticamente serán mejores que los de las fuentes NTP.
- especificando el valor de desfase en segundos, en este caso 0.100183, en el parámetro `time1` (*PPS time offset calibration factor*) [26]. Obsérvese el cambio de signo respecto al valor obtenido en el apartado anterior.

Quedando, por tanto, de la siguiente manera:

```
# GPS PPS via COM12
# mode 49 = 1 + 48 = process $GPRMC + linespeed 38400 bps
server 127.127.20.12 mode 49 minpoll 4 maxpoll 4 prefer
# flag1 = PPS, time1 = offset calibration
fudge 127.127.20.12 time1 +0.100183 flag1 1
```

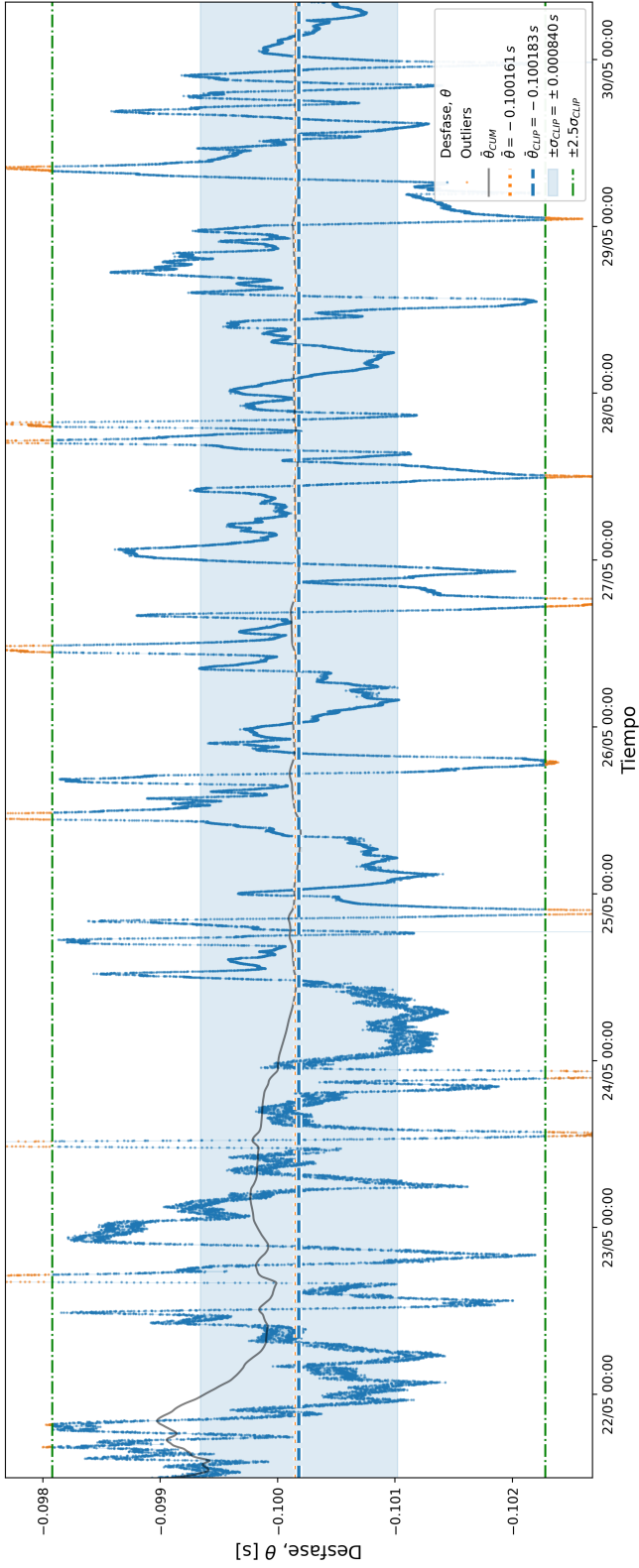


Figura 11. Evolución del desfase de la fuente GPS durante más de 8 días, medido en base a 21 fuentes NTP stratum 1. Prueba realizada en entorno Windows, con Meinberg NTP.

Puede ser interesante reducir el valor de `minpoll` y `maxpoll` a 3, de manera que el sondeo al GPS se realice cada $2^3 = 8$ segundos, si bien en las pruebas realizadas no se han obtenido diferencias significativas.

6. Resultados

Presentamos a continuación los resultados obtenidos tras monitorizar el desfase residual y su desviación estándar, tanto en Linux como en Windows, una vez compensado el desfase estimado en apartados anteriores.

Durante el intervalo de análisis se ha realizado una prueba de carga consistente en una captura de una ROI (*Region of Interest*) de 480 x 480 píxeles para conseguir una tasa de 100 fps (*frames per second*), y comprobar si provoca fluctuaciones en el desfase.

Respecto a la configuración definitiva del servicio NTP, `chrony` o `ntpd`, es conveniente indicar que no es necesario mantener una lista tan exhaustiva de servidores NTP. Basta con mantener unos pocos, y no necesariamente stratum 1, simplemente por disponer de fuentes alternativas en caso de que usemos el equipo de captura sin el receptor GPS.

6.1. Resultados en Linux

Las especificaciones del equipo de captura usado son: Intel Celeron J3455 @ 1.50GHz 4 cores (Q3'16), 4GB RAM, SSD SATA.

Los resultados obtenidos se resumen en la figura 12, que conserva la misma escala vertical que la gráfica de medición de desfase, figura 6, para facilitar su comparación.

Se observa que, tras la corrección de desfase, la media del residuo es inferior a la precisión de medida. Su desviación estándar es de tan sólo 4 μ s, lo que implica una estabilidad excelente para tratarse de una fuente de tiempo transportada a través de USB.

En términos absolutos, se ha apreciado una desviación de hasta 52 μ s, que se produce bajo las condiciones de la prueba de captura y posterior copia del fichero de vídeo a través de red (ethernet). En cualquier caso, este resultado indica una precisión bastante por debajo del milisegundo.

6.2. Resultados en Windows

Las especificaciones del equipo de captura usado son: Intel i7-8750H @ 2.2GHz 6 cores (Q2'18), 32 GB RAM, SSD NVMe.

Los resultados obtenidos se resumen en la figura 13, que conserva la misma escala vertical que la gráfica de medición de desfase, figura 11, para facilitar su comparación. Nótese que dicha escala es diez veces superior a la de los resultados en Linux, 12.

Se observa que, tras la corrección de desfase, la media del residuo es muy pequeña, del orden de microsegundos, o incluso mejor en el caso de sigma-clipping. Su desviación estándar es de 80 μ s, lo que supone una estabilidad bastante aceptable.

En términos absolutos, se ha observado una desviación de hasta 371 μ s, que no se ha podido trazar hasta ninguna actividad concreta en el sistema. Durante las pruebas de captura no se aprecia ningún impacto obvio en el desfase. Se realizaron dos pruebas seguidas para confirmar este aspecto. En cualquier caso, a pesar de que estos resultados no son tan buenos como los obtenidos con Linux, la precisión alcanzada también está holgadamente por debajo del milisegundo.

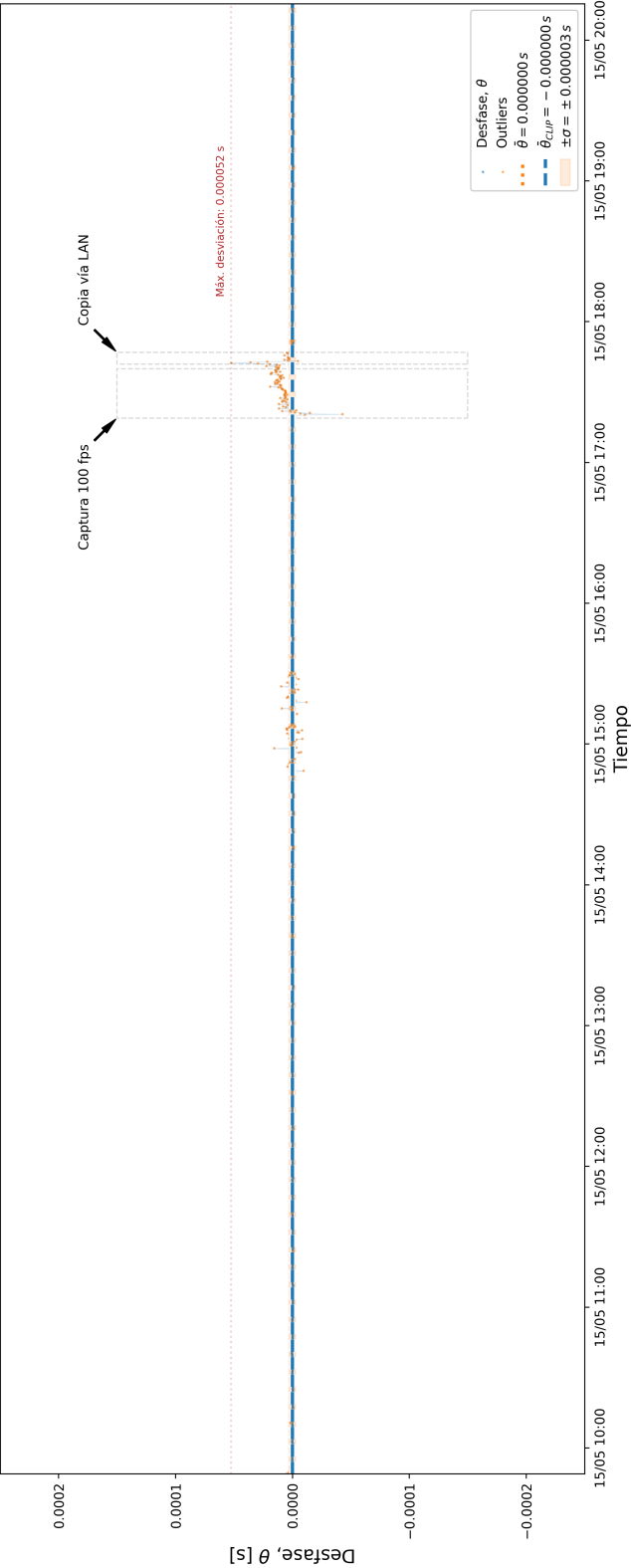


Figura 12. Resultados obtenidos en la prueba realizada en entorno Linux, con *gpsd* y *chrony*.

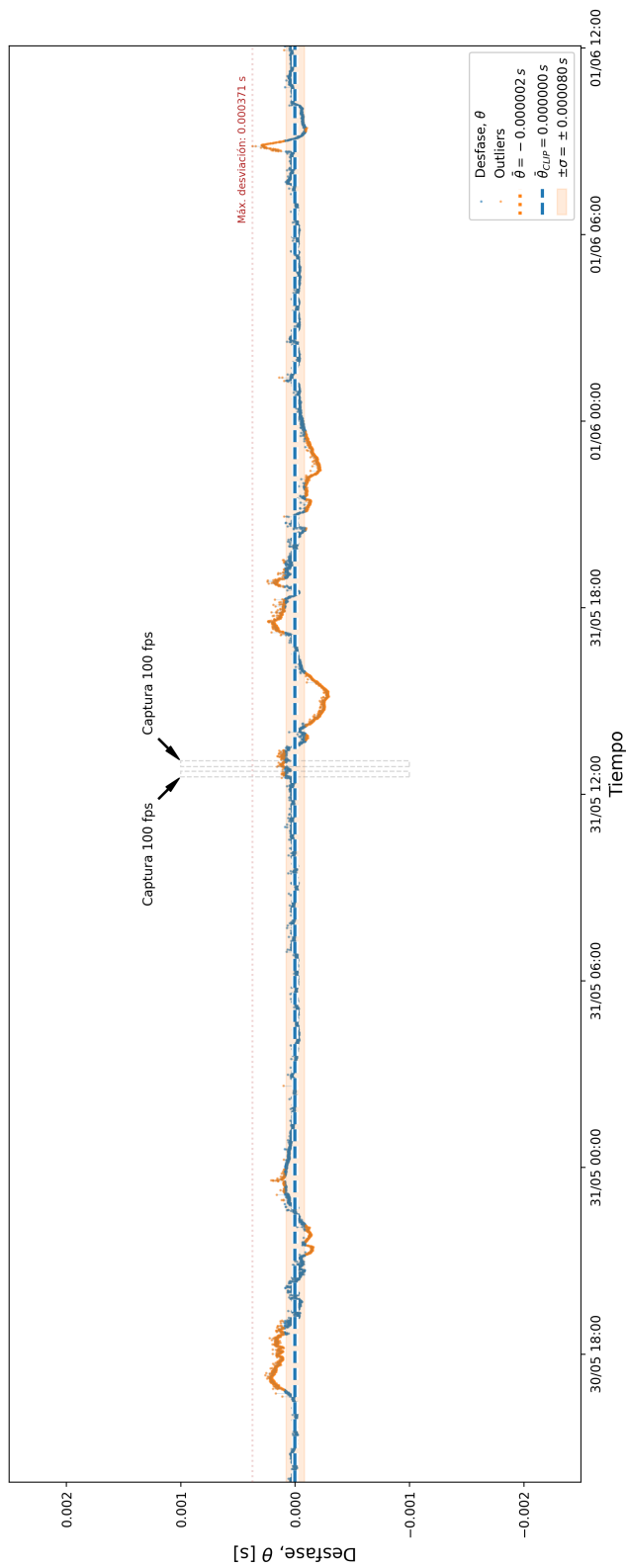


Figura 13. Resultados obtenidos en la prueba realizada en entorno Windows, con Meinberg NTP.

7. Trabajo futuro

Dado que nuestro objetivo final es obtener marcas de tiempo con la mejor precisión que podamos alcanzar en nuestra captura de los fenómenos a observar, debemos considerar también el factor que constituye el software de captura de vídeo, entre los que cabe destacar Sharpcap, Firecapture o AstroDMX Capture.

En un próximo número de esta revista se complementará el presente estudio con un análisis de la latencia instrumental involucrada en este proceso. El experimento propuesto consiste en capturar con una cámara CMOS los destellos del LED PPS incorporado en el receptor GPS, lo que proporcionará una vía alternativa, fotométrica, para determinar los instantes que corresponden al inicio de cada segundo UTC. El estudio desarrollará en particular la problemática asociada a las cámaras con *rolling shutter*.

8. Conclusiones

La solución propuesta para obtener una base de tiempos precisa mediante un módulo GPS a través de USB, según las pruebas realizadas, es capaz de ofrecer la exactitud requerida para las observaciones de ocultaciones estelares, al ser la precisión obtenida inferior al milisegundo. Los factores críticos para lograr esta precisión se resumen en:

- Usar de manera efectiva la señal PPS, que en este caso implica asegurarse de transportarla a través de USB y de configurar el software para asignar adecuadamente a qué instante de tiempo corresponden las marcas temporales que envía el módulo GPS;
- Usar un dispositivo USB-serie cuyo *driver* presente mínima latencia ante cambios en las líneas de control, ya que a través de una de ellas, DCD, se envían los pulsos PPS;
- Calibrar correctamente el desfase asociado a la referencia GPS en el servicio NTP, para lo cual hay que medir su evolución en base a fuentes NTP fiables, y hacer posteriormente un análisis de los datos obtenidos.

En el presente artículo se tratan detalladamente todos los pasos para la implementación de esta solución, incluyendo estos aspectos.

Debe considerarse, no obstante, que el montaje del hardware requiere trabajar con herramientas de electrónica para hacer soldadura. Por otra parte, la medición del desfase puede requerir varios días de captura de datos para obtener valores significativos.

Una cuestión no analizada en este estudio es el impacto que podría tener la activación del modo de ahorro de energía del ordenador portátil sobre la precisión de la base de tiempos.

Referencias

- [1] Repositorio de código en GitHub. <https://github.com/JCAAC-FAAE>
- [2] Beisker et al. Contributions of ‘citizen science’ to occultation astronomy. https://www.researchgate.net/publication/389394183_Contributions_of_'citizen_science'_to_occultation_astronomy. Philosophical Transactions A, The Royal Society. Febrero, 2025.
- [3] George Viscome. Occultation Observing and Recording Primer (v5.1.3). <https://occultations.org/documents/OccultationObservingPrimer.pdf>. International Occultation Timing Association (IOTA). 2024.
- [4] NASA Science Directorate. Uranus Stellar Occultation Campaign 2025. <https://science.larc.nasa.gov/uranus2025/>. Consultada en octubre de 2025.
- [5] David D. Turnell. NASA Lucy Mission. Chasing an Asteroid’s Shadow. <https://lucy.swri.edu/2021/03/17/Occultations.html>. Marzo, 2021.

- [6] Wikipedia. Network Time Protocol. https://en.wikipedia.org/wiki/Network_Time_Protocol. Consultada en octubre de 2025.
- [7] Kevin Sookocheff. How Does NTP Work? <https://sookocheff.com/post/time/how-does-ntp-work/>. Noviembre, 2021.
- [8] Network Time Foundation. How NTP Works. <https://www.ntp.org/documentation/4.2.8-series/warp/>. Noviembre, 2022.
- [9] Comparison of NTP implementations. Chrony Documentation. <https://chrony-project.org/comparison.html>. Consultada en octubre de 2025.
- [10] Wikipedia. Sistema global de navegación por satélite. https://en.wikipedia.org/wiki/Satellite_navigation. Consultada en octubre de 2025.
- [11] Wikipedia. SBAS. <https://es.wikipedia.org/wiki/SBAS>
- [12] Austin Pivarnik. Microsecond accurate NTP with a Raspberry Pi and PPS GPS. <https://austinsnerdythings.com/2021/04/19/microsecond-accurate-ntp-with-a-raspberry-pi-and-pps-gps/>. Abril 2021.
- [13] Austin Pivarnik. Revisiting Microsecond Accurate NTP for Raspberry Pi with GPS PPS in 2025. <https://austinsnerdythings.com/2025/02/14/revisiting-microsecond-accurate-ntp-for-raspberry-pi-with-gps-pps-in-2025/>. Febrero 2025.
- [14] James Clark. PTP Client Hardware. <https://satpulse.net/hardware/clients.html>. Consultada en octubre de 2025.
- [15] James Clark. Windows PTP Client. <https://satpulse.net/howtos/ptp-windows.html>. Consultada en octubre de 2025.
- [16] Módulo Beitian BE-280 basado en u-blox M10050. <https://store.beitian.com/products/beitian-gps-module-with-antenna-ubx-m10050-gnss-chip-ultra-low-power-gnss-receiver-for-track-be-180>
- [17] FTDI. AN232B-04 Data Throughput, Latency and Handshaking. https://ftdichip.com/wp-content/uploads/2020/08/AN232B-04_DataLatencyFlow.pdf. 2006.
- [18] Intel. USB 3.0 Radio Frequency Interference Impact on 2.4 GHz Wireless Devices. <https://www.usb.org/sites/default/files/327216.pdf>. Abril 2012.
- [19] Gary E. Miller, Eric S. Raymond. GPSD Time Service HOWTO. <https://gpsd.gitlab.io/gpsd/gpsd-time-service-howto.html>. Consultada en octubre de 2025.
- [20] chronyc(1) Manual Page. <https://chrony-project.org/doc/4.0/chronyc.html>. Consultada en octubre de 2025.
- [21] chrony.conf(5) Manual Page. <https://chrony-project.org/doc/4.8/chrony.conf.html>. Consultada en octubre de 2025.
- [22] Meinberg NTP Download. <https://www.meinbergglobal.com/english/sw/ntp.htm>. Consultada en octubre de 2025.
- [23] Meinberg NTP Time Server Monitor. <https://www.meinbergglobal.com/english/sw/ntp-server-monitor.htm>. Consultada en octubre de 2025.
- [24] Meinberg Knowledge Base. NTP for Windows: Using PPS Signals on Windows. https://kb.meinbergglobal.com/kb/time_sync/ntp/ntp_for_windows/using_pps_signals_on_windows. Consultada en octubre de 2025.
- [25] u-blox u-center 2. <https://www.u-blox.com/en/u-center-2>. Consultada en octubre de 2025.
- [26] Meinberg Knowledge Base. NTP for Windows: Generic NMEA GPS Receiver. <https://www.meinbergglobal.com/download/ntp/docs/html/drivers/driver20.html>. Consultada en octubre de 2025.