https://farid.ps/articles/rtl9210_usb_to_nvme_bridge/es.html

Linux y el puente USB a NVMe Realtek RTL9210

> **Resumen:**
> • **Síntomas:** Reinicios repetidos de USB, errores de E/S o discos que desaparecen en Linux.
> • **Afectados:** Realtek RTL9210 (confirmado) y RTL9220 (posiblemente).
> • **Causa:** Retroceso al ROM interno (**f0.01**) tras un fallo en la suma de verificación.
> • **Impacto:** Inestabilidad permanente, no hay herramientas de reflasheo para Linux disponibles.
> • **Solución:** Solo las utilidades de Windows de OEM pueden restaurar el firmware; Realtek bloquea alternativas de código abierto.

Preámbulo

En el año 2025, debería ser completamente razonable arrancar un Raspberry Pi desde un SSD conectado por USB. Sin embargo, gracias a las peculiaridades del firmware de Realtek, ese objetivo razonable se ha convertido en una aventura. Tras meses de inestabilidad inexplicable —reinicios aleatorios, discos que desaparecen, sistemas de archivos corruptos — el autor agotó todas las soluciones habituales: cables nuevos, hubs con alimentación, actualizaciones del kernel, ajustes de USB y optimización del firmware. El avance llegó solo cuando ChatGPT respondió a una pregunta extraña a altas horas de la noche: “¿Es posible que el puente USB a NVMe haya revertido a un firmware antiguo?”

Introducción

Si tu carcasa NVMe basada en Realtek se vuelve repentinamente inestable después de semanas de funcionamiento impecable —reinicios repetidos de USB, errores de E/S o discos que desaparecen— no estás solo. Este patrón ha aparecido en varias marcas, desde unidades sin nombre hasta OEM conocidos como Sabrent y Orico. El denominador común: **los chips de puente USB a NVMe Realtek RTL9210 y, posiblemente, RTL9220**.

Al principio, todo funciona. Luego, aparentemente sin causa, el dispositivo comienza a desconectarse bajo carga o durante un uso prolongado, especialmente en sistemas Linux o Raspberry Pi. La verdadera causa no es el SSD ni la fuente de alimentación; es el propio controlador de firmware que retrocede silenciosamente a su **código de respaldo incrustado en ROM**, una versión que Realtek aún envía internamente como **f0.01**.

El mecanismo oculto: Retroceso del firmware por diseño

Los chips de puente de Realtek almacenan su firmware operativo y datos de configuración en un flash SPI externo. Al encenderse, el controlador verifica una suma de verificación simple. Si esa suma no coincide, se niega a cargar el firmware externo y, en cambio, arranca desde su ROM interna.

Ese firmware de respaldo es antiguo y defectuoso. Carece de varias correcciones de estabilidad USB y mejoras en la gestión del estado del enlace presentes en revisiones posteriores, lo que lleva a la secuencia clásica que todo usuario de Linux reconoce:

> usb 3-2: reinicio de dispositivo USB de alta velocidad número 2 usando xhci-hcd
> usb 3-2: lectura del descriptor del dispositivo/64, error -71
> Advertencia EXT4-fs (dispositivo sda2): Error de E/S al escribir en el inodo ...

La suma de verificación puede volverse inválida cuando los datos de configuración se reescriben, por ejemplo, cuando el puente actualiza sus ajustes de gestión de energía o UAS, y el dispositivo pierde energía durante la escritura. El siguiente arranque detecta una suma de verificación corrupta y retrocede permanentemente al firmware de ROM.

En ese momento, tu “carcasa NVMe de alto rendimiento” se comporta exactamente como la carcasa más barata sin marca, porque internamente ahora ejecuta el mismo código base defectuoso grabado en el silicio.

Verificación del problema

Puedes confirmar este estado fácilmente en Linux:

```
lsusb -v | grep -A2 Realtek
```

Un puente Realtek saludable reporta una revisión de firmware (**bcdDevice**) superior a 1.00. Uno que ha retrocedido muestra:

```
bcdDevice f0.01
```

Esa firma **f0.01** significa que el controlador está arrancando desde ROM, y ninguna cantidad de desconexiones, reformateos o ajustes del kernel lo solucionará.

Este mecanismo de retroceso ha sido **confirmado en el RTL9210**. El **RTL9220** parece compartir la misma arquitectura de diseño y disposición del firmware, por lo que podría exhibir un comportamiento idéntico, pero esto sigue siendo **probable, no probado**.

Por qué no puedes repararlo tú mismo

En principio, la solución es trivial: reflashear el firmware correcto al SPI. En la práctica, Realtek hace esto imposible.

La compañía proporciona un actualizador de código cerrado para Windows a OEMs e integradores. A los usuarios de Linux no se les ofrece nada. Los desarrolladores de la comuni-

dad realizaron ingeniería inversa de utilidades de flasheo compatibles (**rtsupdater**, **rtl9210fw**, **rtsupdater-cli**) que permitían la restauración completa del firmware desde sistemas Linux, hasta que Realtek emitió **notificaciones de eliminación por DMCA** para suprimirlas.

No hay una justificación plausible de propiedad intelectual para bloquear tales utilidades: no exponen microcódigo, solo orquestan la secuencia de actualización a través de USB. Las eliminaciones de Realtek no fueron por protección. Fueron ideológicas.

El costo de una ideología

Esto no se trata de idealismo de código abierto. Se trata de la **hostilidad ideológica de un proveedor de hardware hacia sistemas abiertos** que rompe dispositivos comercializados como *compatibles con Linux*.

La resistencia de Realtek a la documentación y las herramientas abiertas ha persistido durante dos décadas, abarcando Wi-Fi, Ethernet, audio y ahora controladores de almacenamiento. Esa insularidad podría pasar desapercibida en un mundo solo de Windows, pero se vuelve tóxica cuando esos mismos chips se integran en productos multiplataforma como el **Sabrent EC-SNVE**, que muestra abiertamente el logotipo de Linux en su empaque.

Al prohibir las utilidades de flasheo para Linux y bloquear el mantenimiento de la comunidad, Realtek ha **criminalizado efectivamente la autorreparación**. Las consecuencias se extienden hacia afuera:

- Los usuarios de Linux ven hardware “soportado” degradarse a la inestabilidad.
- Los OEM como Sabrent y Orico enfrentan costos innecesarios de RMA y garantía.
- La reputación de larga data de Realtek por su mala compatibilidad con Linux se refuerza una vez más.

Al final, no es el código abierto lo que rompe los dispositivos de Realtek; es **la hostilidad de Realtek hacia el código abierto** lo que los rompe.

Un camino racional hacia adelante

La solución no requiere un cambio ideológico, solo pragmatismo. Realtek podría:

1. Lanzar un actualizador de línea de comandos firmado por el proveedor para Linux (no se necesita divulgar el código fuente).
2. Publicar el algoritmo de suma de verificación para que los integradores puedan validar imágenes flash de manera segura.
3. Adoptar un modo estilo DFU que acepte actualizaciones a través de almacenamiento masivo USB, independientemente del sistema operativo.

Cada uno de estos evitaría costos de garantía, protegería las relaciones con los OEM y restauraría la confianza en los chips de puente de Realtek entre los usuarios profesionales de

Linux, desde constructores de estaciones de trabajo hasta desarrolladores de Raspberry Pi.

Qué puedes hacer

Si sospechas que tu carcasa ha retrocedido al firmware de ROM:

- Verifica con **lsusb -v | grep bcdDevice**.
- Si muestra **f0.01**, informa el problema a tu OEM.
- Incluye el extracto de **dmesg** y señala este mecanismo de retroceso documentado.
- Pide a tu proveedor que escale el problema con Realtek, citando la necesidad de un actualizador compatible con Linux.

La política de firmware de Realtek no solo incomoda a los entusiastas; crea pérdidas financieras tangibles para su propio ecosistema. Cuanto antes se reconozca esa realidad dentro de la compañía, antes podrán los usuarios de Linux y los socios OEM dejar de perder tiempo en ciclos de RMA evitables.

Respuestas de los fabricantes

Tanto Realtek como Sabrent fueron invitados a proporcionar declaraciones sobre el problema de retroceso del firmware descrito anteriormente. Sus respuestas, si se reciben, se añadirán aquí.

Apéndice: Identificación de dispositivos afectados

Controlador	ID de proveedor	ID de producto	Notas	Estado
RTL9210	0x0bda	0x9210	Puente USB 3.1 Gen 2 10 Gb/s	**Confirmado** comportamiento de retroceso
RTL9220	0x0bda	0x9220	Puente USB 3.2 Gen 2×2 20 Gb/s	**Posible**, arquitectura similar

Firma de retroceso del firmware: **bcdDevice f0.01**
Revisiones estables conocidas: **1.23** – **1.31**