# Monografía Técnica: Automatización de Despliegue de GSI y Arquitectura de Firmware para Redmi Note 11 Pro 4G (Viva) - Estado del Arte a Diciembre 2025

## 1. Resumen Ejecutivo y Marco Arquitectónico

La integración de Imágenes Genéricas del Sistema (GSI, por sus siglas en inglés) en el Redmi Note 11 Pro 4G, identificado bajo el nombre en clave viva, constituye un caso de estudio paradigmático en el ecosistema contemporáneo de firmware personalizado para Android. A diferencia de los dispositivos impulsados por procesadores Qualcomm Snapdragon, que frecuentemente gozan de una documentación exhaustiva y soporte de kernel "mainline", la plataforma viva opera sobre el chipset MediaTek Helio G96 (MT6781). Esta arquitectura introduce un conjunto distintivo de desafíos y requisitos operativos, que giran primordialmente en torno a la gestión del kernel de los protocolos Berkeley Packet Filter (BPF), la implementación de Project Treble a través de interfaces de proveedor (Vendor Interfaces), y la administración de particiones dinámicas mediante el esquema *super partition*.

El presente informe técnico ofrece un análisis exhaustivo de los mecanismos de instalación automatizada alojados en repositorios de GitHub, gestionados por la organización xiaomi-mt6781-devs y desarrolladores asociados como Woomymy y Emiferpro.1 El análisis deconstruye la lógica de scripting automatizado utilizada para eludir las restricciones inherentes del gestor de arranque (bootloader) de MediaTek, la necesidad crítica de parches de kernel para la compatibilidad con Android 12, 13, 14 y el emergente Android 15/16 hacia finales de 2025, y las intrincadas cadenas de dependencia requeridas para estabilizar entornos basados en AOSP (Android Open Source Project) en este hardware específico. Asimismo, se examina la transición de la interacción manual mediante fastboot hacia el scripting de shell automatizado (flash\_all.sh) y el papel fundamental de mtkclient para la recuperación ante desastres a bajo nivel en el modo Brom.

Hacia diciembre de 2025, el panorama de desarrollo para el viva ha madurado significativamente, con la aparición de ROMs basadas en Android 15 y 16, como LineageOS 23.0 y Rising OS, las cuales dependen intrínsecamente de la infraestructura de automatización consolidada en los años previos para mantener la funcionalidad en un hardware que, oficialmente, ha cesado su ciclo de soporte mayor por parte del fabricante.3

### 1.1 El Paradigma de Hardware: Helio G96 (MT6781)

El Redmi Note 11 Pro 4G utiliza el SoC MediaTek Helio G96. En el contexto del desarrollo de firmware, este chipset dicta los *Device Tree Blobs* (DTB) específicos y las configuraciones de kernel requeridas para un arranque exitoso. El dispositivo se comercializó originalmente con MIUI 13 basado en Android 11, estableciendo un diseño de partición de proveedor específico con el que las GSIs deben interactuar. El árbol de dispositivos para viva, mantenido en GitHub, abstrae estos detalles de hardware —tales como la GPU Mali-G57 MC2 y el Procesador de Señal de Imagen (ISP) especializado para la cámara de 108MP— en un formato compatible con el *Android Common Kernel* (ACK), aunque fuertemente ligado a las ramas legacy 4.14 y 4.19 de Linux.1

La distinción entre las variantes viva (Global) y vida (a menudo citada en registros de compilación o modelos de ingeniería relacionados) es crucial para la selección de archivos de firmware. Los scripts automatizados y mapas de partición discutidos en este documento están específicamente sintonizados para la tabla de particiones de viva, la cual emplea el esquema de Partición Dinámica de Android. Este esquema encapsula las particiones system, vendor, product y odm dentro de una única partición física super, cambiando fundamentalmente cómo los scripts de flasheo deben dirigirse al almacenamiento del dispositivo, obligando el uso de fastbootd (userspace fastboot) para operaciones de escritura lógica.6

### 1.2 Project Treble y el Mandato GSI

Project Treble desacopla la implementación del proveedor (controladores de hardware de bajo nivel) del marco del sistema operativo Android. Idealmente, esto permite que una Imagen Genérica del Sistema arranque en cualquier dispositivo compatible. Sin embargo, la plataforma viva exhibe desviaciones específicas de la implementación estándar, particularmente en su versionado del Kit de Desarrollo Nativo de Video (VNDK) y las configuraciones de overlay requeridas para el brillo de la pantalla y la gestión de energía, las cuales no son nativas en las imágenes GSI puras y requieren inyección post-instalación o superposición en tiempo de ejecución.8

Los scripts automatizados de GitHub analizados en este informe sirven como un puente técnico entre el cumplimiento teórico de Project Treble y la realidad práctica del hardware viva. Automatizan la inyección de banderas en vbmeta para deshabilitar el *Android Verified Boot* (AVB), redimensionan particiones lógicas para acomodar imágenes AOSP más grandes que las stock, y flashean las imágenes de arranque (boot.img) parcheadas necesarias para rectificar la regresión BPF inherente en las series de kernel utilizadas por este dispositivo, un paso no negociable para la conectividad en versiones modernas de Android.10

## 2. El Ecosistema de Desarrollo en GitHub para Viva

El panorama de desarrollo para el Redmi Note 11 Pro 4G está centralizado alrededor de organizaciones específicas de GitHub y repositorios de desarrolladores que alojan el código fuente, los árboles de dispositivos y las herramientas de instalación automatizada. Este ecosistema no es estático; evoluciona con cada iteración de Android, adaptando los scripts para superar nuevas barreras de seguridad y compatibilidad.

### 2.1 La Organización xiaomi-mt6781-devs

El centro neurálgico para el desarrollo de código abierto para el chipset MT6781 es la organización xiaomi-mt6781-devs. Este grupo mantiene varios repositorios críticos que forman la columna vertebral de cualquier instalación de ROM personalizada o GSI en el viva. La estructura modular de estos repositorios permite la colaboración distribuida y la integración continua de parches de seguridad y mejoras de rendimiento.

| **Nombre del Repositorio** | **Función Técnica** | **Criticidad en el Despliegue** |
| --- | --- | --- |
| android\_device\_xiaomi\_viva | Contiene el árbol de dispositivos, configuración de la placa (BoardConfig), y esquemas de partición específicos para viva. | **Crítica** |
| android\_vendor\_xiaomi\_viva | Aloja los *blobs* binarios propietarios extraídos del firmware oficial de MIUI necesarios para la funcionalidad del hardware (cámara, sensores, RIL). | **Crítica** |
| kernel\_xiaomi\_viva | El código fuente del kernel, basado en el kernel MediaTek 4.19, frecuentemente requiriendo parches (backports y security fixes) para nuevas versiones de Android. | **Alta** |
| releases | Un repositorio de almacenamiento para compilaciones finales, zips OTA y, crucialmente, scripts de instalación como flash\_all.sh. | **Media/Alta** |
| android\_device\_xiaomi\_viva-kernel | Pre-compilados del kernel para facilitar la construcción rápida sin recompilar todo el código fuente. | **Media** |

El análisis de los fragmentos de código y actividad revela que desarrolladores como Woomymy y mikeNG son contribuyentes activos al repositorio android\_device\_xiaomi\_viva, realizando commits relacionados con el *Lineage Health HAL*, servicios bluetooth AIDL, y controladores de alarma de apagado.1 Estos commits no son meras actualizaciones de código; representan el "pegamento" lógico que permite que un sistema Android genérico se comunique con el hardware específico del Redmi Note 11 Pro, traduciendo las llamadas del sistema estándar a las instrucciones propietarias que el hardware de MediaTek espera recibir.

### 2.2 El Rol de los Scripts Automatizados y flash\_all.sh

La consulta del usuario resalta específicamente el "script de instalación automatizado GSI en github". En el contexto de viva, esto se refiere a los scripts de shell (frecuentemente nombrados flash\_all.sh, install.sh o incrustados dentro de archivos ZIP flasheables) encontrados dentro de los archivos de lanzamiento o los repositorios de código fuente. La automatización es una respuesta directa a la complejidad de las particiones dinámicas; un error humano al escribir en una partición lógica puede resultar en un *soft-brick* difícil de revertir para el usuario promedio.

Una instancia específica de tal automatización se encuentra en el repositorio releases de xiaomi-mt6781-devs o en las compilaciones no oficiales de LineageOS publicadas por mantenedores como Woomymy.12 Estos scripts están diseñados para abstraer la complejidad de los comandos fastboot. Una instalación manual típica implica más de una docena de comandos individuales para flashear particiones específicas y deshabilitar la verificación. El script automatizado consolida esto en un flujo de ejecución único, manejando la verificación de errores y la detección de ranuras (A/B partitioning) automáticamente.

Por ejemplo, el análisis del flujo de trabajo detallado en 12 y 13 para una "[Unofficial] LineageOS for Redmi Note 11 Pro 4G (viva)" revela un script flash\_all.sh que ejecuta una secuencia crítica de operaciones:

1. **Interacción con el Bootloader:** Asume que el dispositivo está en modo Fastboot y verifica la conectividad.
2. **Gestión de VBmeta:** Aplica banderas --disable-verity y --disable-verification a los comandos de flasheo de vbmeta, un paso obligatorio para evitar que el *Secure Boot* rechace las imágenes no firmadas por Xiaomi.
3. **Transición a FastbootD:** Reconoce la limitación del fastboot estándar para escribir en particiones lógicas y fuerza el reinicio al espacio de usuario (fastboot reboot fastboot).
4. **Flasheo de Particiones Lógicas:** Itera sobre system.img, vendor.img, product.img, gestionando el redimensionamiento o eliminación de particiones redundantes (como product) para liberar espacio en la super partición.

### 2.3 Mantenedores Comunitarios y Releases Activos

Más allá de la organización central, desarrolladores individuales mantienen variantes específicas de ROMs. Emiferpro, por ejemplo, es identificado como el mantenedor de versiones de DerpFest y BananaDroid para viva.14 Estos repositorios a menudo incluyen tipos de compilación "Monthly" (Mensual) y cambios específicos del dispositivo almacenados en archivos como changelog\_viva.txt. La existencia de scripts como createjson.sh y upload.sh en estos repositorios indica un alto nivel de automatización en la Integración Continua/Despliegue Continuo (CI/CD), asegurando que las ROMs proporcionadas a los usuarios se generen consistentemente a partir de las últimas configuraciones de fuente, minimizando la discrepancia entre el código fuente y el binario distribuido.

## 3. Profundización Técnica: La Regresión del Kernel BPF y el Parcheo

Un obstáculo técnico crítico para la instalación de GSI en el Redmi Note 11 Pro 4G es el "problema BPF". Este es un problema sofisticado a nivel de kernel que hace que las GSIs estándar sean inarrancables o funcionalmente deficientes (específicamente en lo que respecta a las redes) en dispositivos MediaTek que ejecutan versiones de kernel más antiguas (4.14/4.19) al intentar arrancar Android 12 o superior.

### 3.1 El Mecanismo eBPF y la Divergencia de MediaTek

El *Extended Berkeley Packet Filter* (eBPF) es una tecnología en el kernel de Linux que permite ejecutar programas en un entorno aislado (sandbox) dentro del kernel sin cambiar el código fuente del kernel ni cargar módulos. Android 12 introdujo requisitos más estrictos para el soporte de eBPF con el fin de manejar la contabilidad del tráfico de red y las políticas de seguridad de manera más eficiente y segura.

Las fuentes del kernel de MediaTek para el MT6781 (y chips similares de esa generación) contienen un commit específico que rompió funcionalidades de BPF, concretamente relacionadas con arraymap, en un intento de corregir errores del *Undefined Behavior Sanitizer* (UBSAN). Esta "corrección" por parte de MediaTek inadvertidamente causó una regresión donde el sistema operativo Android —esperando un comportamiento estándar de BPF— falla al inicializar las interfaces de red, lo que lleva a bucles de arranque (bootloops) o a una falta total de conectividad Wi-Fi y de datos móviles.10

### 3.2 La Automatización con mtk-bpf-patcher

Para resolver esto, la comunidad desarrolló mtk-bpf-patcher, una herramienta basada en Python alojada en GitHub por el desarrollador R0rt1z2.10 Esta herramienta representa un componente crucial del ecosistema de "instalación automatizada" para viva. Sin ella, la viabilidad de cualquier ROM moderna sería nula.

La herramienta opera realizando un parcheo binario en la imagen del kernel compilada (boot.img):

1. **Análisis (Parsing):** El script analiza el binario del kernel para localizar la secuencia de instrucciones específica introducida por el commit defectuoso de MediaTek.
2. **Inyección de NOP:** Reemplaza las instrucciones ofensivas con instrucciones NOP (*No Operation*). Esto elimina efectivamente el código que bloquea la llamada memcpy() requerida para la creación adecuada del mapa BPF.
3. **Reempaquetado:** El kernel parcheado se reempaqueta entonces en una boot.img que puede ser flasheada al dispositivo.

Este proceso a menudo se automatiza dentro de las guías de instalación o scripts proporcionados por los desarrolladores. Por ejemplo, la evidencia en 16 discute el uso de la aplicación Android mtk-bpf-patcher o la herramienta de línea de comandos para generar una imagen de arranque parcheada (e.g., SRPSL24A005...img) que luego se flashea al dispositivo. La automatización de este paso es vital, ya que realizar el parcheo manualmente requeriría conocimientos avanzados de ingeniería inversa y ensamblador ARM64 por parte del usuario final.

### 3.3 Implementación en GitHub

El repositorio mtk-bpf-patcher 10 incluye un setup.py y un script Install.ps1 (PowerShell) para Windows, facilitando la integración sencilla en el flujo de trabajo de los usuarios en diferentes sistemas operativos. La descripción de la herramienta establece explícitamente su propósito: "Herramienta CLI para parchear kernel(s) basados en MTK con el fin de corregir regresión(es) de BPF". Esto confirma que para el Redmi Note 11 Pro 4G, utilizar este script no es opcional, sino un requisito previo obligatorio para la estabilidad funcional de las GSI modernas.

## 4. Arquitectura del Script de Instalación Automatizado

El "script automatizado de GitHub" referenciado en la consulta del usuario es probablemente un compuesto de varios bloques funcionales encontrados en flash\_all.sh o archivos por lotes similares distribuidos con ROMs personalizadas para este dispositivo. A continuación, se deconstruye la arquitectura de tal script para entender exactamente cómo manipula el firmware del dispositivo.

### 4.1 Verificaciones de Requisitos Previos y Entorno

Un script de instalación robusto comienza verificando el entorno del host. Comprueba la presencia de binarios adb y fastboot en el PATH del sistema.

* **Comando:** fastboot devices
* **Lógica:** Si no se devuelve ningún ID de dispositivo, el script se detiene y solicita al usuario que verifique las conexiones USB o los controladores.17
* **Disponibilidad de Controladores:** El script a menudo confía en que el usuario haya instalado los controladores USB de Xiaomi, que son distintos de los controladores USB estándar de Google. Los repositorios de GitHub frecuentemente enlazan o incluyen estos controladores para asegurar la comunicación a bajo nivel.18

### 4.2 El Desarme de Vbmeta (AVB)

El *Android Verified Boot* (AVB) asegura que el código que se ejecuta en el dispositivo provenga de una fuente confiable. Al instalar una GSI personalizada, la firma de la partición del sistema cambia, lo que normalmente desencadenaría un fallo de arranque por seguridad.

* **Comando Crítico:** fastboot --disable-verity --disable-verification flash vbmeta vbmeta.img
* **Análisis:** Este comando es fundamental. La bandera --disable-verity desactiva la función dm-verity (*device-mapper verity*), que comprueba la integridad del dispositivo de bloques. La bandera --disable-verification desactiva la verificación de firma del árbol hash de la partición. La imagen vbmeta.img flasheada aquí es a menudo una imagen "en blanco" o modificada proporcionada en el lanzamiento de GitHub para asegurar que estas banderas sean aceptadas por el bootloader.13 Sin este paso, el dispositivo entraría en un estado de "Red State" o bootloop inmediato.

### 4.3 Gestión de Particiones Dinámicas (FastbootD)

El Redmi Note 11 Pro 4G utiliza particiones dinámicas. El modo fastboot estándar (bootloader) no puede redimensionar o escribir en particiones lógicas dentro del contenedor super. El script debe reiniciar el dispositivo en fastbootd (fastboot en espacio de usuario).

* **Comando:** fastboot reboot fastboot
* **Indicador:** La pantalla del dispositivo cambia del logotipo estándar "FASTBOOT" (o el conejo de Xiaomi reparando el androide) a un menú basado en texto titulado "fastbootd".6

Una vez en fastbootd, el script aborda las restricciones de almacenamiento. Las GSIs son a menudo más grandes que la partición del sistema stock, que está ajustada al byte exacto por Xiaomi para ahorrar espacio.

* **Redimensionamiento de Partición Lógica:**
  + fastboot delete-logical-partition product\_a (o product\_b)
  + fastboot delete-logical-partition product
* **Razonamiento:** La partición super tiene un tamaño físico fijo. Para hacer espacio para una GSI de Sistema grande (a menudo 2GB+), el script a menudo elimina la partición product, que contiene aplicaciones y personalizaciones regionales del operador no necesarias para las GSIs AOSP. Esto libera los bloques necesarios dentro del contenedor super para permitir la expansión de system.13

### 4.4 El Flasheo del Sistema y Formateo

Con el espacio asegurado y la verificación deshabilitada, el script ejecuta la entrega de la carga útil principal.

* **Comando:** fastboot flash system system.img
* **Contexto:** La system.img es la Imagen Genérica del Sistema extraída del archivo descargable (e.g., LineageOS, PixelExperience). El script automatiza la fragmentación (*chunking*) de este archivo grande si se alcanza el límite de imagen dispersa (sparse limit) del protocolo USB.6

Para prevenir conflictos de encriptación y asegurar un entorno limpio, el script realiza un borrado.

* **Comando:** fastboot -w
* **Análisis:** Esto borra la partición userdata. Es esencial porque las claves de encriptación generadas por el sistema operativo anterior (MIUI) no serán compatibles con la nueva GSI (AOSP). No realizar este borrado resulta en un "bootloop" donde el dispositivo se cuelga en la pantalla de animación debido a la imposibilidad de montar /data.12

## 5. Panorama de ROMs GSI Activas para Viva (Diciembre 2025)

Respondiendo a la segunda parte de la solicitud del usuario, el análisis de los repositorios y la actividad de la comunidad hacia finales de 2025 revela un conjunto específico de ROMs GSI que mantienen un desarrollo activo y soporte para el Redmi Note 11 Pro 4G (viva). La "actividad" se define aquí por la frecuencia de actualizaciones, la integración de parches de seguridad de Android recientes, y la compatibilidad con las últimas versiones del sistema operativo (Android 15 y 16).

### 5.1 LineageOS (Versiones 22.0 y 23.0)

LineageOS permanece como el estándar de oro para la estabilidad en el viva.

* **Estado a Dic 2025:** El repositorio de releases de xiaomi-mt6781-devs muestra una actividad sostenida. Se identifican lanzamientos etiquetados como "LineageOS 23.0" en octubre de 2025.3
* **Versión de Android:** LineageOS 22.0 corresponde a Android 15, mientras que LineageOS 23.0 correspondería a Android 16 (o una versión muy avanzada de Android 15 QPR).
* **Características Clave:** Enfoque en la privacidad, ausencia de bloatware, y alta compatibilidad con el hardware gracias a los parches específicos aplicados por mantenedores como Woomymy. Es la base recomendada para usuarios que buscan estabilidad diaria.

### 5.2 RisingOS

RisingOS ha ganado tracción significativa en la comunidad de personalización hacia 2024 y 2025.

* **Estado:** Referencias en videos y guías recientes 4 indican tutoriales de instalación para "Rising OS GSI ROM... 2025".
* **Enfoque:** Personalización extrema de la interfaz de usuario (UI) y optimizaciones de rendimiento ("High FPS Gaming").
* **Compatibilidad:** Utiliza la base de Treble, por lo que su instalación en viva sigue el mismo procedimiento automatizado de scripts detallado anteriormente.

### 5.3 DerpFest y BananaDroid

Mantenidas históricamente por Emiferpro, estas ROMs ofrecen una experiencia cercana a Pixel pero con características adicionales.

* **Estado:** Aunque los registros muestran actividad intensa en 2023 y 2024, la persistencia de estos proyectos depende del mantenedor. Sin embargo, dado que Emiferpro es un desarrollador clave para el árbol de dispositivos viva, es altamente probable que continúen existiendo compilaciones actualizadas o sucesores espirituales bajo su supervisión.14

### 5.4 Android 15 AOSP (GSI Pura)

Para desarrolladores y puristas, las imágenes GSI directas de AOSP o compilaciones de Google.

* **Estado:** Google continúa publicando binarios GSI para pruebas. Hacia diciembre de 2025, las betas de Android 16 o versiones estables de Android 15 QPR2 estarían disponibles.7
* **Advertencia:** Estas imágenes carecen de superposiciones específicas del dispositivo (overlays), por lo que problemas como el brillo o el VoLTE podrían requerir intervención manual intensiva si no se usan scripts de post-instalación adecuados.

**Tabla Comparativa de ROMs Activas (Estimación Dic 2025):**

| **ROM** | **Base Android** | **Nivel de Automatización** | **Enfoque Principal** | **Mantenedor Principal (Ref.)** |
| --- | --- | --- | --- | --- |
| **LineageOS 23.0** | Android 16/15 | Alto (Scripts dedicados) | Estabilidad, Privacidad | xiaomi-mt6781-devs / Woomymy |
| **RisingOS** | Android 15 | Medio (GSI Genérica) | Personalización UI, Gaming | Comunidad Treble / Guías YouTube |
| **DerpFest** | Android 14/15 | Alto (JSONs de actualización) | Pixel+ Features, Fluidez | Emiferpro |
| **Project Infinity** | Android 15 | Medio | Rendimiento | Comunidad Treble 19 |

## 6. Entornos de Recuperación y Automatización: TWRP y OrangeFox

Mientras que los scripts automatizan el proceso vía Fastboot, los *Custom Recoveries* como TWRP (*Team Win Recovery Project*) y OrangeFox Recovery proporcionan una alternativa basada en interfaz gráfica (GUI) y son a menudo instalados *vía* scripts como un paso preliminar o de seguridad.

### 6.1 OrangeFox para Viva

El repositorio device\_xiaomi\_viva-recovery en SourceForge, enlazado desde las páginas de desarrollo de GitHub, aloja las compilaciones de OrangeFox Recovery.20

* **Archivo:** OrangeFox-Unofficial-viva.zip
* **Características Críticas:** Este recovery soporta la desencriptación de datos de MIUI (hasta Android 13/14), permitiendo a los usuarios acceder a su almacenamiento interno incluso si el dispositivo está encriptado. También incluye herramientas para flashear archivos .img directamente a particiones, imitando efectivamente la funcionalidad de los scripts de fastboot pero en el dispositivo.21
* **Automatización:** OrangeFox permite el uso de scripts de "OpenRecoveryScript" para automatizar tareas de flasheo y limpieza (wipe) sin intervención del usuario una vez iniciado el recovery.

### 6.2 Integración con GSI

Las guías de "Woomymy" a menudo sugieren usar el recovery para flashear complementos (*add-ons*) tales como GApps (Google Apps) o Magisk (Root) después de la instalación principal de la GSI. El script de instalación principal podría terminar reiniciando al usuario en modo recovery:

* **Comando:** fastboot reboot recovery
* **Propósito:** Permitir al usuario realizar tareas finales de "limpieza" como redimensionar el sistema de archivos (si la GSI no ocupó todo el espacio de la partición lógica) o instalar correcciones de SafetyNet.12

## 7. Post-Instalación Automatizada: Overlays y Correcciones

La instalación de la GSI es solo la mitad de la batalla en el Redmi Note 11 Pro 4G. Debido a las discrepancias en la Capa de Abstracción de Hardware (HAL), varios componentes (Brillo, Audio, VoLTE) a menudo fallan inicialmente. Los repositorios de GitHub proporcionan overlays (superposiciones) automatizados para corregir esto.

### 7.1 La Corrección del Overlay de Brillo

Un problema común en el viva ejecutando GSIs es el escalado incorrecto del brillo, donde la pantalla permanece demasiado oscura o los controles deslizantes no son lineales.

* **Mecanismo:** La corrección implica un *Runtime Resource Overlay* (RRO). Este es un archivo APK pequeño compilado a partir de recursos que definen los enteros de la curva de brillo (e.g., config\_screenBrightnessSettingMinimum, config\_screenBrightnessSettingMaximum).
* **Fuente en GitHub:** El treble\_overlay\_xiaomi\_viva 8 o los overlays genéricos de Phh-Treble contienen estos valores.
* **Aplicación Automatizada:** Los scripts o instrucciones post-instalación dirigen al usuario a habilitar "Force alternative backlight scale" en la configuración de Phh-Treble, o el mantenedor de la ROM (Emiferpro) incluye el APK del overlay en los parches de la partición vendor antes de compilar la imagen final.13

### 7.2 Magisk y Automatización de Root

El acceso Root es a menudo requerido para aplicar modificaciones profundas del sistema que los overlays no pueden tocar.

* **MagiskBoot:** Herramientas como magiskboot se utilizan para desempaquetar la boot.img, parchear el kernel (potencialmente aplicando la corrección BPF simultáneamente si no se hizo antes), y reempaquetarlo.
* **Integración de Script:** Usuarios avanzados utilizan scripts que combinan el parcheador BPF y el parcheo de Magisk en un flujo de trabajo único. El script flash\_all.sh en el fragmento 12 alude al flasheo de imágenes boot.img que ya pueden estar pre-parcheadas por el mantenedor, simplificando la experiencia del usuario final.

## 8. Recuperación ante Desastres: La Herramienta MTKClient

Ningún análisis técnico de la instalación de firmware en MediaTek está completo sin abordar el "unbricking" (desbrickeo). La herramienta mtkclient alojada en GitHub es la red de seguridad definitiva para los usuarios de viva.

### 8.1 Explotación del Modo Brom

Si un flasheo de GSI defectuoso corrompe el preloader o la partición de arranque, el dispositivo puede dejar de entrar en modo Fastboot. Sin embargo, el chipset MT6781 contiene una Memoria de Solo Lectura de Arranque (Brom) que se activa puramente por conexión de hardware (inserción USB mientras se mantienen presionados botones específicos, usualmente Volumen Arriba + Volumen Abajo).

* **Herramienta:** mtkclient (desarrollada por bkerler y contribuidores).23
* **Función:** Utiliza exploits (como Kamakiri o vulnerabilidades de desbordamiento de búfer en el handshake USB) para eludir la autenticación DAA (Download Agent Authentication) usualmente requerida por la cuenta de servicio autorizada de Xiaomi.
* **Capacidad:** Permite acceso completo de lectura/escritura a todas las particiones (incluyendo boot, system, vendor, nvram, seccfg) directamente desde la línea de comandos de Python.
* **Uso en Automatización:** Aunque no es típicamente parte del script de *instalación* diario, mtkclient es el script de *restauración*. Un usuario puede ejecutar un comando simple como:  
  Bash  
  python mtk.py w boot,vbmeta boot.img,vbmeta.img  
    
  Esto restaura instantáneamente el dispositivo a un estado arrancable utilizando imágenes stock descargadas de repositorios oficiales como XiaomiROM 24, revirtiendo efectivamente cualquier daño causado por scripts GSI defectuosos o incompatibilidades de versión.

## 9. Conclusiones y Perspectivas Técnicas

El ecosistema de instalación automatizada de GSI para el Redmi Note 11 Pro 4G (viva) es un ensamblaje sofisticado de scripting de shell, parcheo de kernel basado en Python y gestión de particiones dinámicas. Está impulsado por la necesidad de superar las limitaciones específicas del chipset MediaTek Helio G96 —específicamente la regresión BPF y la estricta implementación de AVB—.

La organización de GitHub xiaomi-mt6781-devs sirve como el repositorio central para el árbol de dispositivos y las fuentes del kernel, permitiendo a desarrolladores como Woomymy producir compilaciones de LineageOS que son casi indistinguibles de las ROMs nativas en términos de estabilidad. Los scripts flash\_all.sh encapsulan la complejidad de las interacciones con FastbootD, asegurando que incluso usuarios no familiarizados con el redimensionamiento de particiones lógicas puedan desplegar sistemas basados en AOSP con éxito.

Hacia diciembre de 2025, la relevancia de estas herramientas solo ha aumentado. A medida que Android evoluciona hacia las versiones 15 y 16, la divergencia entre los kernels legacy 4.14/4.19 y los requisitos "mainline" de Android se ensancha, haciendo que la herramienta mtk-bpf-patcher y los scripts de automatización de overlays sean componentes no solo útiles, sino estructuralmente necesarios para la supervivencia del dispositivo en el ecosistema de software moderno. La combinación de automatización en fastbootd y recuperación vía mtkclient proporciona una vía robusta, aunque técnica, para extender la vida útil del viva mucho más allá de su soporte oficial.

## 10. Referencia Técnica Detallada

Las siguientes secciones proporcionan datos granulares sobre los comandos, estructuras de partición y lógica de script analizados anteriormente.

### 10.1 Estructura de la Tabla de Particiones

El dispositivo viva utiliza el esquema de Particiones Dinámicas de Android. La partición física super actúa como un contenedor para múltiples particiones lógicas, cuyo tamaño puede variar dinámicamente siempre que la suma no exceda el tamaño de super.

| **Nombre de Partición** | **Tipo** | **Contenido** | **Modo de Flasheo Requerido** |
| --- | --- | --- | --- |
| boot | Física | Kernel, Ramdisk | Bootloader / Fastboot |
| dtbo | Física | Device Tree Overlay | Bootloader / Fastboot |
| vbmeta | Física | Verified Boot Metadata | Bootloader / Fastboot |
| super | Física | Contenedor | Bootloader / Fastboot (Solo lectura/flasheo completo) |
| system | Lógica | OS Framework (Objetivo GSI) | Userspace Fastboot (fastbootd) |
| vendor | Lógica | Drivers, HALs | Userspace Fastboot (fastbootd) |
| product | Lógica | Apps regionales, personalización | Userspace Fastboot (fastbootd) |
| userdata | Física | Datos de usuario, claves de encriptación | Bootloader / Fastboot |

### 10.2 Lógica de Instalación "Woomymy" / LineageOS

Basado en el análisis de 12, la lógica de instalación sigue una secuencia estricta para asegurar la integridad de la partición y la capacidad de arranque.

Bash

# 1. Reiniciar al Bootloader  
adb reboot bootloader  
  
# 2. Flashear Recovery (Opcional pero recomendado para sideloading)  
fastboot flash boot boot.img  
# Nota: En algunas ROMs, el recovery está empaquetado dentro de boot.img  
  
# 3. Deshabilitar AVB (Crucial para GSI)  
fastboot --disable-verity --disable-verification flash vbmeta vbmeta.img  
fastboot --disable-verity --disable-verification flash vbmeta\_system vbmeta\_system.img  
fastboot --disable-verity --disable-verification flash vbmeta\_vendor vbmeta\_vendor.img  
  
# 4. Reiniciar a FastbootD (Userspace)  
fastboot reboot fastboot  
  
# 5. Flashear Particiones Lógicas  
# El script determina la ranura activa (A o B) si aplica, aunque viva es usualmente A-only en lógica dinámica  
fastboot flash system system.img  
fastboot flash product product.img # A menudo una imagen vacía (dummy) para ahorrar espacio  
fastboot flash vendor vendor.img # Solo si se actualiza el firmware del proveedor  
  
# 6. Borrar Datos (Format)  
fastboot -w  
  
# 7. Reiniciar  
fastboot reboot

### 10.3 Lógica Interna de mtk-bpf-patcher

El mtk-bpf-patcher es el eje para las redes en Android 12+. Sin él, la GSI arranca, pero el demonio netd falla repetidamente.

Desglose Técnico:

El parcheador apunta al código del compilador JIT (Just-In-Time) del kernel.

* **Problema:** Un commit en la fuente del kernel MTK modifica cómo se asignan los mapas BPF, causando un fallo de comprobación de límites en el cargador BPF estándar de Android.
* **Solución:** El script de Python escanea el binario boot.img (después de desempaquetarlo para obtener el archivo kernel). Busca la secuencia de bytes hexadecimal específica que corresponde a la comprobación restrictiva.
* **Parche:** Sobrescribe esta secuencia con NOPs (0x00 o NOP específico de la arquitectura ARM64).
* **Resultado:** Cuando el SO intenta cargar un mapa BPF, la comprobación se salta, la memoria se asigna, y las redes se inicializan correctamente.

Esta herramienta se ejecuta *antes* de flashear la boot.img. En los scripts automatizados proporcionados en los lanzamientos de GitHub, la boot.img dentro del archivo ZIP usualmente ya ha sido procesada por esta herramienta por el mantenedor (Woomymy o Emiferpro), relevando al usuario final de este paso complejo.

#### Obras citadas

1. Device tree for the redmi note 11 pro 4g (viva) - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/xiaomi-mt6781-devs/android_device_xiaomi_viva>
2. xiaomi-mt6781-devs repositories - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/orgs/xiaomi-mt6781-devs/repositories>
3. xiaomi-mt6781-devs/releases: LineageOS Releases - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/xiaomi-mt6781-devs/releases>
4. How to Install GSI ROM on Any Android | Rising OS ... - YouTube, fecha de acceso: diciembre 28, 2025, <https://www.youtube.com/watch?v=YX29sHmQVco>
5. Device Tree for Redmi Note 11S (4G) - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/xiaomi-mt6781-fleur-dev/android_device_xiaomi_fleur>
6. Install Android GSI on Redmi Note 9 Series - Medium, fecha de acceso: diciembre 28, 2025, <https://medium.com/@fahim-ahmed/how-to-install-android-11-gsi-on-redmi-note-9-pro-61d56965b9ed>
7. Android 15 GSI binaries and release notes - Android Developers, fecha de acceso: diciembre 28, 2025, <https://developer.android.com/about/versions/15/gsi-release-notes>
8. Xiaomi MT6781 Development - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/xiaomi-mt6781-devs>
9. Brightness at half when 100% on Honor 8X Max on Any GSI (AB) Rom, fecha de acceso: diciembre 28, 2025, <https://github.com/phhusson/treble_experimentations/issues/1204>
10. R0rt1z2/mtk-bpf-patcher - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/R0rt1z2/mtk-bpf-patcher>
11. setup.py - R0rt1z2/mtk-bpf-patcher - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/R0rt1z2/mtk-bpf-patcher/blob/main/setup.py>
12. [Unofficial] LineageOS for Redmi Note 11 Pro 4G (viva) : r/androidroot, fecha de acceso: diciembre 28, 2025, <https://www.reddit.com/r/androidroot/comments/1llyt3i/unofficial_lineageos_for_redmi_note_11_pro_4g_viva/>
13. Flashing GSI on Xiaomi Mi 11T Pro (vili) treble phhusson - GitHub Gist, fecha de acceso: diciembre 28, 2025, <https://gist.github.com/danpawlik/acb2f59d370fce55bfe0a3b92eee78b8>
14. Emiferpro/releases: derp releases for viva - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/Emiferpro/releases>
15. Emiferpro/releases - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/Emiferpro/releases/releases>
16. Unofficial GSI Android 15 - Page 2 - e/OS community, fecha de acceso: diciembre 28, 2025, <https://community.e.foundation/t/unofficial-gsi-android-15/71674?page=2>
17. Download Xiaomi Flash Tool, fecha de acceso: diciembre 28, 2025, <https://www.xiaomiflash.com/>
18. Download Xiaomi Mi Flash Tool, fecha de acceso: diciembre 28, 2025, <https://mipcsuite.com/mi-flash-tool/>
19. Redmi Note 11: Installing Latest Android 15 - Project Infinity OS, fecha de acceso: diciembre 28, 2025, <https://www.youtube.com/watch?v=50Y0PbRZE84>
20. device\_xiaomi\_viva-recovery download | SourceForge.net, fecha de acceso: diciembre 28, 2025, <https://sourceforge.net/projects/device-xiaomi-viva-orangefox/>
21. Download OrangeFox-Unofficial-viva.zip (device\_xiaomi\_viva ..., fecha de acceso: diciembre 28, 2025, <https://sourceforge.net/projects/device-xiaomi-viva-orangefox/files/OrangeFox-R11-1.1-viva-20072023-Beta-Stable/OrangeFox-Unofficial-viva.zip/download>
22. Install Orange Fox Recovery on Redmi Note 11 Decrypt ... - YouTube, fecha de acceso: diciembre 28, 2025, <https://www.youtube.com/watch?v=2gTvO-5UXD8>
23. bkerler/mtkclient: MTK reverse engineering and flash tool - GitHub, fecha de acceso: diciembre 28, 2025, <https://github.com/bkerler/mtkclient>
24. MIUI 14 Global Stable ROM v14.0.3.0.TGDMIXM with Android 13 for ..., fecha de acceso: diciembre 28, 2025, <https://new.c.mi.com/global/post/617599>
25. Redmi Note 11 Pro 4G (viva) Global Fastboot & Recovery ROM, fecha de acceso: diciembre 28, 2025, <https://xiaomirom.com/en/rom/redmi-note-11-pro-4g-viva-global-fastboot-recovery-rom/>