



UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Memoria de Título

Programa para la transmisión de señal ISDB-T en PC usando tarjeta moduladora DTA-115 y sistema operativo Linux

Memoria para optar al título de
Ingeniero Civil Electrónico

Alumno

Emilio Enrique Ascencio Henríquez

Profesor Supervisor

Agustín José González Valenzuela

Revisores/Comisión Co-Referente

Gonzalo Andrés Carvajal Barrera

3 de noviembre de 2025, Valparaíso, Chile



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: "Programa para la transmisión de señal ISDB-T en PC usando tarjeta moduladora DTA-115 y sistema operativo Linux"

Nombre del candidato(a): Emilio Enrique Ascencio Henríquez

Carrera / Grado: Ingeniero Civil Electrónico

Campus: Casa Central **Departamento:** Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Agustín J. González Valenzuela, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 03.11.2025

Firma:

Estudiante o Candidato(a):

Fecha: 03/11/2025

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

AGRADECIMIENTOS

Quiero comenzar expresando mi más sincero agradecimiento a mi familia, quienes siempre han sido un pilar fundamental en todo el proceso académico que he recorrido. Su apoyo incondicional me ha acompañado en cada paso de este camino, siendo su confianza y aliento mi mayor motivación para seguir adelante en este proceso.

A mis amigos, tanto de la universidad, el voluntariado, Casa Beta, que también han jugado un papel esencial en mi vida, les agradezco profundamente. Ellos hicieron que la universidad no solo fuera un lugar de aprendizaje, sino también un espacio de crecimiento personal y apoyo mutuo. Gracias a su compañía y a las experiencias compartidas, mis años universitarios fueron mucho más llevaderos y enriquecedores.

A mi pareja, Irene Larral, quien ha sido durante varios años un pilar fundamental en mi vida. Su presencia ha sido constante, brindándome el apoyo que necesitaba tanto en los momentos de éxito como en los de incertidumbre. Su amor y comprensión han sido esenciales para lograr este objetivo, y no tengo palabras suficientes para agradecerle por todo lo que ha hecho por mí en este proceso.

A mis queridas mascotas, Ginny, Mary, Annie y Chely, les debo un agradecimiento especial. Su presencia constante me ha brindado una compañía incondicional y un apoyo silencioso, pero invaluable durante este proceso. Han sido una fuente de energía, convirtiéndose en un pilar emocional en los momentos de estrés y en los días de descanso. Gracias por estar siempre ahí, aportando con su amor incondicional.

*Trabajo dedicado a Enrique Henríquez Gutierrez,
mi padre; quien durante toda su vida se
esforzó al máximo para que todos sus hijos
salieran adelante y que no les faltara nada.
Te amo papá y espero que estés orgulloso.*

Programa para la transmisión de señal ISDB-T en PC usando tarjeta moduladora DTA-115 y sistema operativo Linux

Emilio Enrique Ascencio Henríquez

Memoria para optar al título de Ingeniero Civil Electrónico

Mención en Computadoras

Profesor Guía: Agustín José González Valenzuela

Octubre 2025

Resumen

La televisión digital es una tecnología clave en la difusión de contenido audiovisual, y con el crecimiento de las normas de transmisión, el estándar **ISDB-T** se ha consolidado como uno de los más utilizados por su flexibilidad y robustez. Sin embargo, en Linux falta una solución integrada y accesible para el proceso completo bajo ISDB-T, lo que aumenta la complejidad y limita su aplicación; se privilegia Linux debido a que es el entorno más extendido, de menor costo y alta disponibilidad. En este contexto, la presente memoria tiene como objetivo desarrollar un programa integrado y accesible en Linux para la transmisión de señal de televisión mediante el protocolo **ISDB-T**, utilizando la tarjeta moduladora DekTec DTA-115.

El proyecto se enfoca en la implementación eficiente de la codificación, modulación y transmisión de contenido audiovisual a través de **ISDB-T**. Para ello, se desarrolló un software en lenguaje Python que interactúa con la tarjeta moduladora, permitiendo una transmisión estable y confiable. Además, se realizaron pruebas exhaustivas para verificar la calidad de la señal transmitida y su estabilidad en diversas condiciones.

Los resultados obtenidos demuestran que el sistema propuesto es una solución viable y efectiva para la transmisión de señales de televisión bajo el estándar **ISDB-T**, contribuyendo al desarrollo de tecnologías de transmisión de televisión digital más eficientes y accesibles.

Palabras clave: *ISDB-T, transmisión de señal, televisión digital, tarjeta DekTec, Python.*

Program for the Transmission of ISDB-T Television Signals on PC Using DTA-115 Modulator Card and Linux Operating System

Emilio Enrique Ascencio Henríquez

Final Project Report towards the fulfillment of the Ingeniero Civil Electrónico
Minor in Computer Systems degree

Advisor: Agustín González

October 2025

Abstract

Digital television is a key technology for distributing audiovisual content, and with the growth of transmission standards, the **ISDB-T** protocol has become one of the most widely used for its flexibility and robustness. However, on Linux there is still no integrated and accessible solution for the complete process under **ISDB-T**, which increases complexity and limits its application; Linux is favored because it is the most widespread, lower-cost, and highly available environment. In this context, this thesis aims to develop an integrated and accessible Linux program for the transmission of television signals using the **ISDB-T** protocol with the DekTec DTA-115 modulator card.

The project focuses on the efficient implementation of encoding, modulation, and transmission of audiovisual content through **ISDB-T**. To this end, software was developed in Python that interacts with the modulator card, enabling stable and reliable transmission. In addition, exhaustive tests were conducted to verify the quality of the transmitted signal and its stability under diverse conditions.

The results show that the proposed system is a viable and effective solution for television signal transmission under the **ISDB-T** standard, contributing to the development of more efficient and accessible digital television transmission technologies.

Keywords: **ISDB-T**, *signal transmission, digital television, DekTec card, Python*

Glosario

16-QAM 16 Symbols - Quadrature Amplitude Modulation.

1seg One Segment.

64-QAM 64 Symbols - Quadrature Amplitude Modulation.

HD Alta Definición, por sus siglas en inglés **H**igh **D**efinition.

ISDB-T Integrated Services Digital Broadcasting – Terrestrial.

OFDM Orthogonal Frequency Division Multiplexing.

QPSK Quadrature Phase Shift Keying.

RF Radiofrecuencia.

SD Definición Estándar, por sus siglas en inglés **S**tandard **D**efinition.

SDK Software Development Kit.

TDA Televisión Digital Abierta.

TDT Televisión Digital Terrestre.

ÍNDICE

RESUMEN EJECUTIVO	6
ABSTRACT	7
GLOSARIO	8
1. INTRODUCCIÓN	11
1.1. Marco teórico	12
1.1.1. Términos clave	12
1.2. Contexto Nacional	16
1.3. Descripción del problema	17
1.4. Solución propuesta	18
1.5. Público objetivo	21
2. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DEL	
PROGRAMA	22
2.1. Entorno de desarrollo	22
2.2. Herramientas ya existentes para la transmisión de TVD con tarjetas Dektec	24
3. DISEÑO DE ODISEASTREAM	26
3.1. Visión general y arquitectura del sistema	26
3.2. Modelo de procesos y flujos de datos	29
3.3. Componentes y responsabilidades	31
3.4. Decisiones de diseño y justificación	33
3.5. Casos de uso operativos	36
	9

4. EVALUACIÓN DE ODISEASTREAM	39
4.1. Interfaz gráfica	39
4.2. Generación archivo .ts	40
4.3. Previsualización local	48
4.4. Transmisión por tarjeta moduladora DTA-115	50
4.4.1. Un canal	51
4.4.2. Dos canales	52
4.5. Matriz de trazabilidad	53
5. CONCLUSIONES	55
5.1. Trabajo futuro	56
BIBLIOGRAFÍA	57
ANEXOS	60

INTRODUCCIÓN

La televisión digital terrestre ha revolucionado la manera en que se transmiten y consumen los contenidos audiovisuales. Con el avance de los estándares de transmisión, el protocolo **ISDB-T** (Integrated Services Digital Broadcasting – Terrestrial) se ha establecido como uno de los más utilizados en varios países de América Latina y Asia, incluyendo Chile [\[Subsecretaría de Telecomunicaciones, 2009\]](#). Este estándar se caracteriza por su robustez, flexibilidad y capacidad de adaptarse a diferentes condiciones geográficas, permitiendo la transmisión eficiente de señales de televisión digital.

En el contexto del desarrollo tecnológico actual, la implementación de programas capaces de gestionar la transmisión de estas señales de manera eficiente ha adquirido una importancia crucial. Este trabajo se enmarca en el diseño y desarrollo de un programa en Linux para la transmisión de señal de televisión digital utilizando el estándar **ISDB-T**, con la tarjeta moduladora DekTec DTA-115. El objetivo principal es implementar una solución eficiente y confiable que permita la transmisión de contenidos a través de este protocolo, garantizando una correcta modulación y codificación de la señal.

El enfoque de esta memoria es la transmisión de la señal de televisión, excluyendo funcionalidades adicionales como la interacción con la audiencia o el canal de retorno. Este ajuste permite concentrarse en la optimización del proceso de transmisión en sí, lo cual es

fundamental para garantizar la estabilidad y calidad del servicio.

Para lograr los objetivos planteados, se desarrolló un programa en lenguaje Python llamado *OdiseaStream*, diseñado específicamente para interactuar con la tarjeta DekTec DTA-115, gestionando la transmisión de señales de televisión digital bajo el estándar **ISDB-T**. A lo largo de este trabajo se presentarán los fundamentos teóricos del sistema de transmisión, las características del protocolo **ISDB-T**, la implementación del software, y los resultados de las pruebas realizadas para evaluar el rendimiento del sistema en diversas condiciones.

El desarrollo de este proyecto no solo busca cumplir con los requisitos técnicos de transmisión, sino que también pretende contribuir al avance de tecnologías que puedan mejorar la accesibilidad y la calidad de la televisión digital en Chile y en otros países que utilizan este estándar.

1.1. Marco teórico

A continuación, se establecen los fundamentos conceptuales para comprender la televisión digital terrestre y, en particular, la operación bajo la norma ISDB-T adoptada en Chile. Se precisan los términos clave y componentes del sistema, destacando las propiedades de robustez y flexibilidad que posibilitan servicios en distintos escenarios de recepción. Asimismo, se delimita la adaptación regional ISDB-Tb y sus implicancias para la implementación local. Este marco proporciona el vocabulario técnico y los parámetros que sustentarán el diseño, la configuración y la evaluación del software orientado a la tarjeta moduladora DTA-115.

1.1.1. Términos clave

- **Televisión Digital Terrestre (TDT)**: También conocida como **Televisión Digital Abierta (TDA)** es la transmisión secuencial de imágenes para simular movimiento y su sonido asociado, enviados en codificación binaria a través de una red de transmisores terrestres. **[Consejo Nacional de Televisión de Chile, 2011]**

La transmisión digital presenta ventajas frente a la transmisión analógica, como: el uso más eficiente del espectro radioeléctrico, ya que es capaz de transmitir mediante

multiplexación más de una señal televisiva; capacidad de transmisión de audio y video de alta definición (HD), en comparación a la señal analógica que transmite en calidad estándar (SD); la multiplexación también permite gestionar el ancho de banda de forma tal, que puede emitir más de un flujo de audio, más de un flujo de subtítulos o información asociada al programa que se está transmitiendo.

- **Estándar ISDB-T**: El estándar ISDB-T (Integrated Services Digital Broadcasting-Terrestrial) fue desarrollado en Japón para la transmisión de televisión digital terrestre sobre canales de 6 MHz, con arquitectura segmentada (13 segmentos) que posibilita servicios fijos y móviles en un mismo multiplex. Sobre esta base, varios países de América Latina –entre ellos Chile– adoptaron la variante conocida como ISDB-Tb o ISDB-T International, una adaptación regional impulsada y diseñada en Brasil en el marco de sus Sistema Brasileño de TV Digital (SBTVD). Esta adaptación mantiene la estructura técnica esencial del sistema y la compatibilidad de recepción, pero introduce ajustes para responder a necesidades locales.

En términos funcionales, ISDB-Tb incorpora:

- Mayor eficiencia de compresión al estandarizar MPEG-4/AVC (H.264) para video y HE-AAC para audio, mejorando la relación calidad/tasa de bits respecto a la configuración original.
- Plataforma de interactividad mediante el middleware Ginga, orientada a aplicaciones interactivas y servicios complementarios.
- Compatibilidad plena con 13 segmentos, incluyendo el servicio 1-Seg para recepción portátil, preservando la filosofía de "servicios integrados" del estándar.

Para el caso chileno, la elección de ISDB-Tb se sustenta en:

- (1) Eficiencia espectral y calidad de servicio con tasas de bits contenidas.
- (2) Interoperabilidad regional, que favorece disponibilidad de receptores y contenidos.
- (3) Alineamiento regulatorio con la normativa técnica nacional para TDT, que establece parámetros de compresión, audio y servicios complementarios consistentes con esta variante.

■ **Características técnicas del protocolo ISDB-T**

1. **Modulación OFDM:** Modulación que divide la señal en múltiples subportadoras que se transmiten de forma simultánea, lo que proporciona resistencia a interferencia y a los desvanecimientos de la señal causados por la propagación multicamino; además el uso de modulación OFDM es lo que permite la transmisión simultánea de distintos tipos de servicios a través del mismo canal de 6 MHz.
2. **Configuración de capas jerárquicas:** La transmisión jerárquica de ISDB-T permite enviar múltiples flujos de datos en forma simultánea dentro de un mismo canal de 6 MHz, asignando distintas partes del multiplex OFDM (segmentos) a capas con parámetros diferentes de modulación y corrección de errores. De este modo, cada capa puede operar con robustez y capacidad propias; una capa más robusta para recepción móvil (1-Seg) y otras con mayor tasa de bits para servicios fijos (SD/HD).

Esta configuración optimiza el uso del espectro y permite adaptar la calidad y la cobertura según las características del dispositivo receptor.

3. **Flexibilidad en la Modulación:** ISDB-T soporta diversos esquemas de modulación, tales como QPSK, 16-QAM, 64-QAM, lo que permite ajustar la relación entre la capacidad de datos y la robustez de la señal en función de las necesidades del entorno de transmisión. En áreas con alta interferencia, puede usarse un esquema de modulación más robusto como QPSK, mientras que en entornos con menos interferencia se puede usar 64-QAM para maximizar la capacidad de datos.
4. **Sistema de corrección de errores:** Se utiliza un avanzado sistema de corrección de errores basado en los códigos Reed–Solomon [Association of Radio Industries and Businesses (ARIB), 2014] y la codificación convolucional ($k=7; 1/2-7/8$) [Association of Radio Industries and Businesses (ARIB), 2014], combinados con entrelazados byte/bit/tiempo/frecuencia [Digital Broadcasting Experts Group (DiBEG), 2007], que protege los datos transmitidos contra las distorsiones y pérdidas de información, garantizando una recepción más fiable incluso en condiciones adversas.

5. **Interactividad y servicios múltiples:** El estándar tiene la capacidad de soportar servicios interactivos en la medida que un dispositivo receptor disponga de un canal de retorno, lo que permite la comunicación bidireccional entre los receptores y los proveedores de contenido. Esto facilita aplicaciones como la votación en tiempo real o la compra de productos directamente desde el televisor.
 6. **Segmentación para dispositivos móviles:** El estándar también permite la transmisión de una señal dedicado a dispositivos móviles conocido como **1seg**, lo que lo diferencia de otros estándares de televisión digital. 1Seg permite que teléfonos móviles y otros dispositivos portátiles reciban la señal de televisión en baja resolución (320x240 píxeles) brindando acceso a la televisión en tiempo real en cualquier lugar, sin necesidad de una conexión a Internet.
- **Tarjeta DekTec DTA-115:** Es un modulador de televisión digital de alto rendimiento que permite la generación y transmisión de señales bajo una variedad de estándares de transmisión terrestre, cable y satélite (ISDB-T, DVB-T, DVB-C, QAM, ATSC y 8VSB). **DekTec Digital Video BV, 2011**

Características técnicas principales:

1. **Modulación digital:** La tarjeta soporta esquemas de modulación como QPSK, 16-QAM y 64-QAM así como 8VSB para ATSC. Estos esquemas permiten adaptar la transmisión según la calidad y la robustez requerida, asegurando una transmisión eficiente en función de las condiciones del entorno.
2. **Frecuencia de operación:** La tarjeta cubre un amplio rango de frecuencias, desde 36 MHz hasta 2150 MHz, lo que la hace adecuada para transmitir señales tanto en el rango de frecuencia utilizado en la televisión terrestre como en la televisión por cable.
3. **Interfaz PCI:** La tarjeta se conecta a un puerto PCI en el computador, lo que facilita su integración en sistemas basados en Linux o Windows. Esta interfaz proporciona un alto rendimiento en la transmisión de datos, lo cual es crucial para aplicaciones que requieren la modulación y transmisión de señales en tiempo real.
4. **Software de desarrollo:** La tarjeta es compatible con un conjunto de herramientas de software proporcionadas por DekTec, que permite a los

desarrolladores controlar todos los aspectos de la modulación y la transmisión de la señal.

1.2. Contexto Nacional

En Chile, el proceso de implementación de la televisión digital terrestre (TDT) ha sido gradual, pero sostenido, desde la promulgación de la Ley N°20.750 [Gobierno de Chile, 2014](#) en mayo de 2014, la cual establece el marco normativo para la introducción de la televisión digital abierta bajo el estándar ISDB-Tb. Esta legislación marca un hito en la transición tecnológica del país, reemplazando el sistema analógico tradicional por una alternativa más eficiente que permite una mejor calidad de imagen, un mayor número de señales por frecuencia, movilidad y nuevas funciones interactivas.

El estándar ISDB-Tb, adoptado oficialmente por Chile, es una variante del sistema japonés adaptado para las necesidades latinoamericanas por Brasil, y ha sido seleccionado por la mayoría de los países de la región. Este modelo no solo contempla la transmisión de señal de alta definición, sino que también incorpora servicios complementarios, como subtítulos, múltiples canales de audio, guías electrónicas de programación (EPG) y aplicaciones interactivas.

En este escenario, han surgido diversas iniciativas desde universidades, centros de investigación y organismos públicos para fortalecer el conocimiento y la infraestructura asociada a esta nueva tecnología. Instituciones como la Subsecretaría de Telecomunicaciones (SUBTEL) han liderado campañas informativas y proyectos de expansión de cobertura, especialmente en zonas rurales y de difícil acceso.

Sin embargo, a nivel formativo, aún persisten desafíos significativos relacionados con la capacitación de profesionales y la creación de herramientas didácticas que permitan experimentar y comprender los aspectos técnicos detrás de la transmisión digital. En este sentido, el desarrollo de plataformas como OdiseaStream cobra relevancia, ya que permite simular entornos reales de transmisión a baja escala, utilizando hardware de bajo costo y software de código abierto, fomentando así el aprendizaje práctico y la investigación aplicada en el contexto nacional.

1.3. Descripción del problema

La transición de la televisión analógica a la televisión digital terrestre (TDT) ha representado un cambio profundo en la forma de transmitir, recibir y comprender las señales de televisión. Si bien en Chile este proceso está respaldado normativamente y ha avanzado en términos de infraestructura y cobertura, persiste una importante brecha en el acceso a herramientas que permitan experimentar, desarrollar o entender a nivel técnico el funcionamiento de un sistema de transmisión digital.

En contextos académicos y de investigación, el estudio de la norma ISDB-Tb suele quedar limitado al análisis teórico o al uso de software desarticulado: un conjunto de utilidades independientes que deben coordinarse manualmente (configuraciones dispersas, parámetros de servicio y señalización SI/PSI, manejo de consola y múltiples pasos no integrados). Esta situación eleva la complejidad operativa, exige experiencia específica y aumenta la probabilidad de errores de configuración, desincentivando la experimentación y la innovación.

A pesar de la disponibilidad de hardware especializado, como tarjetas moduladoras compatibles con ISDB-Tb, no existen soluciones locales integradas en Linux que permitan controlar de forma simple y autónoma el proceso completo: preparación del contenido, generación del flujo de transporte digital, creación de tablas de señalización SI/PSI y transmisión efectiva por aire. Esta ausencia limita el potencial de formación técnica en este campo.

Existe en Windows un programa que concentra varias de estas funciones (StreamXpress). Sin embargo, su carácter propietario, la ausencia de código fuente y su dependencia de un entorno distinto de Linux limitan la continuidad del desarrollo, la extensión de funcionalidades y la replicabilidad en contextos donde se privilegia Linux por su amplia disponibilidad y menor costo operativo.

En consecuencia, se vuelve necesario desarrollar herramientas que integren, automaticen y simplifiquen el proceso de transmisión digital bajo ISDB-Tb en Linux, habilitando su uso tanto con fines pedagógicos como como base para aplicaciones profesionales o comunitarias.

1.4. Solución propuesta

Frente a la ausencia de herramientas integradas de código abierto que permitan experimentar de manera autónoma con el proceso completo de transmisión digital bajo la norma ISDB-Tb, se propone el desarrollo de un sistema denominado OdiseaStream, cuyo objetivo principal es facilitar la generación y transmisión de flujos digitales utilizando la tarjeta moduladora DTA-115 de DekTec, a través de una interfaz gráfica intuitiva y funcional.

OdiseaStream permite al usuario cargar archivos de video, realizar su conversión a los formatos requeridos para la transmisión digital, generar las tablas de señalización necesarias (PAT, PMT, SDT, NIT, EIT), sincronizar el flujo de transporte resultante y, finalmente, transmitirlo de forma efectiva por aire utilizando la tarjeta moduladora. Todo este proceso se lleva a cabo de manera automatizada, sin necesidad de interacción directa con la línea de comandos ni conocimientos avanzados en codificación o multiplexado.

Una de las características clave de la solución es su interfaz gráfica, desarrollada completamente en Python con Tkinter, a la cual se le añadió el reproductor VLC, que permite visualizar el contenido audiovisual antes y durante la transmisión, configurar los parámetros técnicos de forma accesible y monitorear el estado del proceso en tiempo real. Además, el sistema permite generar flujos con uno o dos canales digitales, lo que amplía su aplicabilidad tanto en entornos formativos como en escenarios experimentales. Un diagrama del programa se puede apreciar en la figura [1.1](#).

Esta solución integra herramientas de software libre, lo que permite su portabilidad, adaptabilidad y distribución sin restricciones de licenciamiento. En particular, la interfaz se implementa con Tkinter (módulo estándar de Python para GUI) [\[Python Software Foundation, 2025\]](#); la previsualización audiovisual se realiza con VLC mediante los bindings `python-vlc` [\[VideoLAN, 2025\]](#), [\[VideoLAN / Olivier Aubert, 2025\]](#); la preparación y normali-

zación de audio y video se efectúa con FFmpeg [FFmpeg Project, 2025a][FFmpeg Project, 2025b]; y la generación de PES/TS, multiplexado CBR, estampado de tiempos y la creación de tablas PSI/SI (PAT, PMT, SDT, NIT, EIT) se apoyan en utilidades de OpenCaster y su librería dvbobjects [Avalpa Digital Engineering, 2025][Debian, 2013].

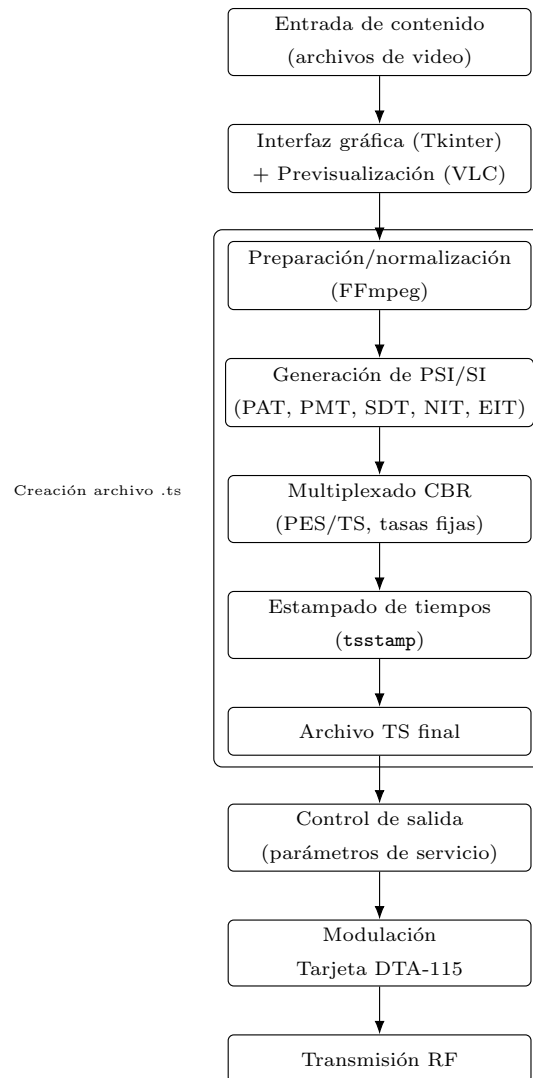


Figura 1.1: Diagrama de OdiseaStream

Por lo tanto, considerando los objetivos propuestos al comienzo del trabajo de memoria, se tiene lo siguiente:

- **Objetivos generales:** Implementar un programa escrito en C/C++ en un computador personal (PC) utilizando las librerías de Linux que permita transmitir una señal de video utilizando el protocolo ISDB-T a través de una tarjeta moduladora DTA-115 de DekTec.
- **Objetivos específicos:**
 - Como entrada, se debe permitir video desde un archivo o desde un flujo de video proveniente de otra máquina.
 - La interfaz permitirá previsualizar los videos a transmitir e, idealmente, conmutar entre dos fuentes de video.

1.5. Público objetivo

OdiseaStream está dirigido a unidades académicas y de formación técnica (universidades, institutos profesionales y liceos técnicos) y a laboratorios docentes que operan en Linux y disponen (o tienen acceso) a tarjetas moduladoras DekTec DTA-115 bajo ISDB-Tb. Estos entornos requieren una herramienta integrada y de bajo costo que automatice la preparación del contenido, la señalización PSI/SI y el multiplexado, y que permita transmitir sin depender de conocimientos avanzados de línea de comandos. La interfaz gráfica de OdiseaStream satisface esta necesidad al guiar el flujo completo de manera reproducible y trazable, facilitando prácticas, evaluaciones y demostraciones en clase.

También se orienta a radios/comunidades educativas, organizaciones locales y colectivos culturales que, por restricciones presupuestarias o de infraestructura, privilegian Linux y software libre para prototipar o emitir contenidos en ISDB-Tb con recursos limitados. En estos casos, OdiseaStream ofrece una vía accesible para generar uno o dos servicios digitales y verificar su recepción en equipos reales, manteniendo la compatibilidad con el hardware docente disponible (DTA-115) y facilitando la replicabilidad del proceso.

Forman parte del público objetivo quienes: (i) utilizan Linux como entorno operativo preferido; (ii) necesitan reducir la complejidad operativa evitando la consola de comandos; (iii) demandan una secuencia guiada y un registro del proceso para fines didácticos o de validación; y (iv) valoran un bajo costo y la continuidad del desarrollo mediante software libre. Bajo estas condiciones, OdiseaStream responde directamente a consolidar en una sola aplicación el flujo completo de transmisión digital ISDB-Tb con DTA-115.

HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DEL PROGRAMA

A continuación, se presenta el conjunto de plataformas y utilidades que hicieron posible el desarrollo de OdisaStream, desde el sistema operativo y el entorno de programación hasta las herramientas de procesamiento audiovisual, multiplexado y control del hardware de modulación. El criterio de selección fue asegurar la estabilidad en Linux, la compatibilidad con ISDB-Tb y con la tarjeta DekTec DTA-115, y la posibilidad de automatizar un flujo reproducible: preparación y normalización de audio/video, generación del Transport Stream y de las tablas SI/PSI, previsualización local y transmisión por RF.

2.1. Entorno de desarrollo

El entorno de desarrollo se definió para asegurar la estabilidad en Linux, la compatibilidad con ISDB-Tb y la tarjeta DekTec DTA-115, así como la automatización reproducible del flujo de trabajo, desde la preparación de contenidos hasta la transmisión RF. Se utilizó Ubuntu 20.04.6 LTS instalado en un disco externo dedicado, a fin de aislar dependencias, facilitar la replicación en otros equipos y mantener controladas las versiones de los compo-

nentes del sistema (kernel, controladores y bibliotecas).

Sistema Operativo y lenguaje:

- Sistema Operativo: Ubuntu 20.04.6 LTS (64 bits); Se privilegia Linux por su bajo costo de adopción, su amplia disponibilidad en laboratorios y la facilidad de réplica del entorno.
- Instalación: Dedicada en disco duro externo USB 3.1.
- Lenguaje de programación: Python 3.8 [Python Software Foundation, 2020](#) y Python 2.7 [Python Software Foundation, 2020](#); siendo la versión de Python 3.8 utilizada para la programación principal de OdiseaStream, que incluye la interfaz gráfica, la lógica de conversión de archivos, el control de flujo y la ejecución de comandos del sistema. Mientras que la versión de Python 2.7 se utiliza exclusivamente para los scripts de generación de tablas SI/PSI. los cuales utilizan la biblioteca dvbobjects de OpenCaster [Debian, 2013](#)

Interfaz de usuario y previsualización local:

- GUI: Tkinter. Módulo estándar de Python para construir interfaces gráficas ligeras y portables sin dependencias externas adicionales. Se utiliza porque está disponible por defecto, ofrece una curva de aprendizaje simple y facilita una distribución estable en entornos Linux de laboratorio. [Python Software Foundation, 2025](#)
- Reproductor: python-vlc. Se utiliza ya que permite reproducir y controlar contenido audiovisual dentro de la aplicación, ofrece una previsualización integrada confiable y evita frameworks pesados o reproductores externos, además de permitir la selección de flujo de video dentro de un archivo .ts [VideoLAN, 2025](#), [VideoLAN / Olivier Aubert, 2025](#)

Herramientas para la preparación, generación y transmisión de Transport Stream:

- FFmpeg: Herramienta multimedia utilizada para la preparación y normalización de las fuentes audiovisuales antes de su incorporación al flujo de transporte [FFmpeg Project, 2025a](#), [FFmpeg Project, 2025b](#)

- OpenCaster (con parche LIFIA para ISDB-Tb): Se utiliza para construir y organizar flujos MPEG-TS con su señalización (PSI/SI) y realizar un multiplexado controlado y reproducible; el entorno se instala con OpenCaster 2.4 y el patch LIFIA (rev362) documentado en guías académicas y trabajos técnicos. [Avalpa Digital Engineering, 2025], [Universidad Técnica Federico Santa María, 2022], [Moncayo Flores, 2014]
- TSDuck: toolkit para el análisis y verificación de Transport Streams; se utiliza la herramienta tstable para comprobar la estructura del servicio y la presencia/consistencia de las tablas PSI/SI. [TSDuck Project, 2025a], [TSDuck Project, 2025b]
- SDK de DekTec (versión 2024.04.0): Proporciona los drivers y binarios necesarios para la tarjeta moduladora DTA-115, incluyendo DtPlay, el software responsable de iniciar la transmisión de los archivos .ts. [DekTec Digital Video BV, 2024]

2.2. Herramientas ya existentes para la transmisión de TVD con tarjetas Dektec

Streamxpress - DekTec

StreamXpress es un software desarrollado por DekTec para el sistema operativo Windows que permite la reproducción y transmisión de archivos Transport Stream (TS) a través de interfaces de hardware de modulación digital desarrolladas por la misma empresa, como las tarjetas DTA-115, DTA-2115B o los dispositivos USB DTU-315. Está diseñado para realizar pruebas, demostraciones, emisiones temporales y verificación de contenido MPEG-2 TS, incluyendo soporte para la norma ISDB-T e ISDB-Tb, así como DVB-T, DVB-C, ATSC, y otros estándares digitales. Sin embargo, se encuentra limitado en funciones de edición; StreamXpress es, por diseño, un software de playout: reproduce en tiempo real archivos ya preparados hacia salidas DekTec (ASI/RF/IP/SDI, etc.). No es un editor, codificador ni multiplexor; no realiza transcodificación ni autoría/edición de PSI/SI, y asume que el Transport Stream viene preconstruido con la estructura y tablas requeridas. Las modificaciones posibles se centran en parámetros de salida y control del modulador, no en la edición del contenido del TS. [DekTec Digital Video BV, 2025]

Este programa forma parte del ecosistema de aplicaciones de prueba de DekTec, utilizado ampliamente en laboratorios, universidades, broadcasters y centros de desarrollo de productos digitales de televisión.

Ventajas:

- Alta compatibilidad con hardware profesional.
- Interfaz gráfica intuitiva.
- Amplio soporte de estándares.
- Control detallado de transmisión.
- Ideal para la validación del flujo TS.

Contras:

- Licencia de pago.
- Limitado en funciones de edición.

DISEÑO DE ODISEASTREAM

A continuación, se presenta el diseño de OdiseaStream, detallando cómo se organiza el sistema para previsualizar, construir y transmitir contenidos bajo el estándar ISDB-Tb utilizando la tarjeta moduladora DTA-115 en Linux. El capítulo expone la arquitectura por capas, el modelo de procesos y flujos de datos, los componentes y sus responsabilidades, y las decisiones de diseño que priorizan la modularidad, la compatibilidad y la trazabilidad del flujo completo (desde la preparación A/V y la señalización SI/PSI hasta la modulación). Asimismo, se describen los casos de uso operativos que guían la interacción del usuario con la interfaz, el reproductor embebido, el asistente de creación de TS y el control de transmisión, junto con las validaciones y los registros que aseguran una operación reproducible y diagnosticable. El código final resultante del diseño se encuentra en *Anexo: Archivo OdiseaStream.py*

3.1. Visión general y arquitectura del sistema

El sistema implementa un flujo integrado para previsualizar, construir y transmitir contenidos audiovisuales bajo ISDB-Tb utilizando una tarjeta DTA-115 en Linux. La aplicación se organiza en capas bien definidas:

- Capa de interfaz y experiencia de usuario (GUI)
 - Interfaz gráfica construida con Tkinter: Ventana principal, barra de controles

(abrir medio, previsualizar, iniciar transmisión, detener), selector de programa y panel de registro.

- Reproducción embebida mediante python-vlc: el reproductor se incrusta en un widget de la GUI, lo que permite el monitoreo local (play, tiempo transcurrido, barra de progreso y volumen).
- Gestión de estados visibles para el usuario: Idle, previsualización y transmisión. El botón “Detener” unifica el cierre seguro de cada modo.
- Panel de logs integrado: Muestra en tiempo real el registro de libVLC y de la aplicación (colas, acotación de tamaño y tail del archivo de log), para facilitar la trazabilidad y el diagnóstico.

■ Capa de orquestación de medios

- Apertura de archivos: Se aceptan TS y contenedores comunes de audio/video. Si el medio es TS, el sistema habilita directamente la transmisión.
- Selección de programa (TS): Se inspecciona el Transport Stream para detectar el número de canales disponibles y se ofrece su elección desde la GUI (etiquetados como “Canal N” para el usuario); también se admite el ingreso manual del ID.
- Sincronización en transmisión: Al iniciar la salida **RF**, la aplicación arranca primero la reproducción embebida y desencadena la modulación con un desfase controlado, de modo que ambas salidas mantengan coherencia perceptual. El modo loop reinicia solamente el reproductor embebido cuando corresponde.

Capa de construcción del Transport Stream (asistente “Crear TS”)

- Asistente por pestañas:
 - Fuentes: Selección de uno o dos archivos de entrada y la carpeta de destino.
 - Parámetros A/V y PIDs: Configuración de resolución, FPS, relación de aspecto y tasas; validación de PIDs (rango 32–8191 y no repetidos entre servicios).
 - Identificadores SI/PSI: Nombres de red, servicio y descripciones para uno o dos programas, con indicador visual de progreso.
- Pipeline determinista (alto nivel):

- Preparación/normalización A/V con la herramienta ffmpeg.
 - Construcción del TS con utilidades específicas para encapsular flujos elementales, multiplexar en CBR y estampar tiempos mediante las herramientas de OpenCaster.
 - Señalización SI/PSI: generación de PAT/PMT/SDT/NIT/EIT (uno o dos servicios) mediante scripts externos.
 - Verificación: Inspección del TS resultante para confirmar la presencia de programas y tablas mediante la herramienta de TSDuck.
 - Realimentación a la GUI: El archivo final se carga automáticamente para previsualizar o transmitir, y se actualiza el selector de programa.
- Capa de control de transmisión
 - Lanzamiento de la modulación mediante la utilidad del fabricante (DtPlay) con parámetros configurables (tipo/índice de tarjeta, frecuencia canalizada, tamaño de paquete, bitrate TS y modo de transmisión).
 - Cierre controlado: La aplicación gestiona la terminación del proceso de transmisión y el estado del reproductor embebido para evitar artefactos (esperas, kill/terminate, limpieza de recursos).
- Consideraciones de plataforma y concurrencia
 - Linux/X11: Inicialización segura de hilos gráficos y preferencia por XCB para la salida de video.
 - Hilos y temporizadores en la GUI: Actualización periódica de progreso/tiempos, seek interactivo sólo en previsualización y coordinación de acciones diferidas.
 - Trazabilidad: Todos los comandos externos y eventos relevantes se registran con prefijos en el panel de logs y en la consola.
- Entradas, salidas y validaciones clave
 - Entradas: uno o dos archivos audiovisuales y parámetros (A/V, PIDs, nombres de red/servicio).
 - Salidas: archivo TS final listo para transmisión y reproducción embebida.
-

- Validaciones: rango y no repetición de PIDs, detección del número de programa en TS, control de errores en procesos externos y retroalimentación visual del avance y el resultado.

3.2. Modelo de procesos y flujos de datos

Esta sección describe los procesos de alto nivel que componen el sistema y cómo fluye la información entre ellos. Se enfatiza el *qué* hace cada proceso, sus entradas/salidas y validaciones, sin detallar implementación.

■ Ingesta y previsualización

- Propósito: Cargar el contenido y permitir una revisión rápida dentro de la interfaz.
- Entradas: Archivo audiovisual (TS o contenedor común).
- Salidas: Medio listo para reproducción embebida; si es TS, lista de programas disponibles para selección.
- Validaciones: Existencia/legibilidad del archivo; detección del número de programa cuando corresponde.
- Notas: Control de tiempo y volumen; *seek* habilitado sólo en previsualización, no durante la transmisión.

■ Preparación de contenido (A/V)

- Propósito: Normalizar las fuentes de audio y video a formatos y parámetros compatibles con la construcción del flujo de transporte.
- Entradas: Uno o dos archivos de origen; parámetros A/V (resolución, *fps*, relación de aspecto y tasas).
- Salidas: Flujos elementales de audio y video listos para encapsulado.
- Validaciones: Reporte de progreso y manejo de errores de la herramienta de preparación; retroalimentación visual en la GUI.

■ Señalización SI/PSI

- Propósito: Generar la señalización mínima requerida por receptores ISDB-Tb.

- Entradas: Identificadores y descriptores (nombre de red, servicios, descripciones) y cantidad de servicios (1 o 2).
 - Salidas: Tablas PAT, PMT, SDT, NIT y EIT coherentes con los servicios definidos.
 - Validaciones: Ejecución en el entorno compatible; verificación posterior con una herramienta de inspección de TSDuck.
- Construcción del *Transport Stream* (TS)
 - Propósito: Integrar audio, video y señalización en un MPEG-TS de tasa constante.
 - Entradas: Flujos elementales A/V, tablas SI/PSI y parámetros de multiplexado.
 - Salidas: TS intermedio y TS final con *timestamps* consistentes.
 - Validaciones: Comprobación de estructura del servicio y presencia de tablas; notificación de errores y progreso en la interfaz.
 - Gestión de identificadores (PIDs)
 - Propósito: Asegurar la consistencia de los identificadores de flujo.
 - Entradas: PIDs propuestos para audio/video y servicios.
 - Reglas: Rango 32–8191 y no repetición entre servicios.
 - Salidas: Conjunto de PIDs validado para su uso en la construcción del TS.
 - Errores: Ante colisiones o valores fuera de rango, la interfaz informa y bloquea la continuación.
 - Control de transmisión
 - Propósito: Emitir el TS final por RF mediante la tarjeta de modulación.
 - Entradas: Archivo TS, parámetros de salida (tipo/índice de tarjeta, frecuencia/canal, tamaño de paquete, *bitrate* y modo).
 - Salidas: Señal modulada en el canal seleccionado.
 - Ejecución coordinada: Se inicia primero la previsualización embebida y, con un retardo controlado, la modulación.
 - Cierre: Terminación ordenada del proceso de transmisión y liberación de recursos.
 - Registro y trazabilidad
-

- Propósito: Facilitar el diagnóstico y la reproducibilidad.
 - Entradas: Eventos de la aplicación y de las herramientas externas.
 - Salidas: Registro en archivo y panel de *logs* con prefijos de categoría; limitación de tamaño y vaciado periódico para la estabilidad.
- Resumen I/O del flujo
- Datos de entrada: Archivos de usuario (1 o 2), parámetros A/V, identificadores SI/PSI, PIDs y parámetros de transmisión.
 - Artefactos intermedios: Flujos A/V normalizados, señalización SI/PSI, TS intermedio.
 - Salida principal: TS final y, opcionalmente, señal RF transmitida.
 - Metadatos/estado: Lista de números de programa, progreso de tareas, *logs* y estado de la GUI (idle / previsualización / transmisión).

3.3. Componentes y responsabilidades

El sistema se organiza en componentes que colaboran a través de artefactos de datos (archivos A/V, ES/PES/TS, tablas SI/PSI) y señales de control (eventos de interfaz, progreso y estado). A continuación, se describen sus responsabilidades y las principales interfaces entre ellos.

- Interfaz de usuario (GUI)
- Presenta la ventana principal, los controles de operación (abrir, previsualizar, crear TS, iniciar, detener, seleccionar programa) y el panel de registros.
 - Emite eventos de usuario (clics, selección de archivos/parámetros) y refleja el estado global del sistema (idle, previsualización, transmisión).
 - Muestra el progreso y los resultados de las tareas largas, y comunica errores recuperables al usuario.
- Reproductor embebido
- Incrusta la reproducción local del medio dentro de la GUI para monitoreo (tiempo, barra de progreso y volumen).

- Opera en modo de previsualización y durante la transmisión; el *loop* afecta exclusivamente a la reproducción local.
 - Expone su estado (reproduciendo/pausado/finalizado) para coordinar acciones del sistema.
- Gestor de medios
 - Abre archivos de entrada (TS u otros contenedores A/V) y verifica la legibilidad.
 - Si el medio es TS, inspecciona la estructura y detecta los programas disponibles para habilitar la selección en la GUI.
 - Publica metadatos útiles (duración, lista de programas, ruta de salida seleccionada).
 - Motor de preparación A/V
 - Normaliza el contenido de audio y video a formatos y parámetros compatibles con la construcción del transporte.
 - Reporta el progreso y los errores; entrega los flujos elementales listos para encapsulado.
 - Opera de forma desacoplada, permitiendo su sustitución sin afectar al resto del sistema.
 - Generador de señalización (SI/PSI)
 - Produce las tablas mínimas requeridas (PAT, PMT, SDT, NIT, EIT) para uno o dos servicios según los identificadores definidos por el usuario.
 - Valida el entorno necesario para su ejecución y entrega los artefactos de señalización al multiplexor.
 - Constructor del *Transport Stream*
 - Encapsula los flujos elementales, integra la señalización y construye el TS a tasa constante, aplicando estampado de tiempos.
 - Emite un TS final apto para transmisión y genera un resumen de verificación (estructura del servicio y presencia de tablas).

- Validador de identificadores
 - Comprueba que los PIDs definidos sean numéricos, estén en el rango 32–8191 y no se repitan entre los servicios.
 - Bloquea la continuación ante colisiones o valores fuera de rango y entrega mensajes claros a la GUI.
- Control de transmisión
 - Inicia la modulación de la tarjeta de salida con los parámetros seleccionados (tipo/índice de tarjeta, canal/frecuencia, tasa, modo).
 - Coordina el desfase entre la previsualización local y el arranque de la transmisión para mantener la consistencia perceptual.
 - Gestiona el cierre ordenado del proceso de transmisión y la liberación de recursos.
- Registro y diagnóstico
 - Centraliza el registro de eventos de la aplicación y de herramientas externas en un archivo y en un panel de *logs*.
 - Limita el tamaño de los buffers, aplica rotación/recorte y etiqueta mensajes por categoría para facilitar la trazabilidad.
- Interfaces entre componentes (datos y control)
 - Datos: archivos de usuario (entradas), flujos elementales normalizados, tablas SI/PSI, TS intermedio, TS final.
 - Control: eventos de GUI (inicio/pausa/detener, crear TS, seleccionar programa), notificaciones de progreso/errores, y estados del reproductor y de transmisión.
 - Post-proceso: resumen de verificación del TS y actualización automática de la GUI (selector de programa, disponibilidad de transmisión).

3.4. Decisiones de diseño y justificación

Se resumen las decisiones técnicas clave adoptadas y su justificación en términos de portabilidad, compatibilidad con ISDB-Tb, simplicidad operativa y trazabilidad del proceso.

- Plataforma y despliegue
 - **Linux (Ubuntu 20.04.6 LTS):** Disponibilidad en laboratorios, estabilidad y facilidad para reproducir el entorno en equipos heterogéneos (control de versiones del kernel, drivers y herramientas de línea de comandos).
 - **Aislamiento en disco externo:** Evita interferir con instalaciones preexistentes y facilita la clonación del entorno para docencia o demostraciones.
- Interfaz de usuario y previsualización
 - **Tkinter:** Módulo estándar de Python, sin dependencias externas pesadas; permite una GUI ligera y portable adecuada para contextos con recursos limitados [Python Software Foundation, 2025].
 - **python-vlc:** Enlaces a *libVLC* para previsualizar y controlar el contenido embebido en la propia ventana, reduciendo la fricción (no se abre un reproductor externo) y manteniendo el control desde la aplicación [VideoLAN, 2025] [VideoLAN / Olivier Aubert, 2025].
- Preparación de contenido audiovisual
 - **FFmpeg:** Herramienta madura y multiplataforma para normalizar fuentes A/V antes de su incorporación al flujo de transporte; su CLI facilita automatizar tareas y registrar exactamente lo ejecutado [FFmpeg Project, 2025a] [FFmpeg Project, 2025b].
- Construcción del *Transport Stream* y señalización
 - **OpenCaster (con parche LIFIA):** Utilidades en Linux para construir y organizar flujos MPEG-TS y su señalización (PSI/SI), favoreciendo un multiplexado CBR controlado y reproducible; el parche LIFIA asegura compatibilidades prácticas para ISDB-Tb utilizadas en el entorno académico [Avalpa Digital Engineering, 2025] [Universidad Técnica Federico Santa María, 2022] [Debian, 2013].
 - **Separación de etapas:** Preparación A/V, generación de tablas PAT/PMT/SDT/NIT/EIT, multiplexado CBR y estampado de tiempos; esta modularidad facilita el diagnóstico y el reemplazo de componentes, si es necesario.

- Verificación y diagnóstico
 - **TSDuck (tstables)**: Herramienta de inspección para comprobar la estructura del servicio y la presencia/consistencia de las tablas PSI/SI en el TS resultante [TSDuck Project, 2025a, TSDuck Project, 2025b].
 - **Registro centralizado**: Panel de *logs* en la GUI (y archivo) con prefijos de categoría; permite trazar errores/progreso y reproducir acciones.
- Control de transmisión y hardware
 - **DTA-115 + DtPlay (LinuxSDK)**: Compatibilidad directa con el flujo TS y control explícito de parámetros de salida (tarjeta, canal/frecuencia, tasa); disponibilidad local y documentación del fabricante [DekTec Digital Video BV, 2011, DekTec Digital Video BV, 2024].
 - **Sincronización operativa**: La previsualización embebida se inicia primero y la modulación se dispara con un retardo breve para mantener coherencia perceptual entre ambas salidas.
- Validaciones y experiencia de usuario
 - **PIDs**: Verificación de rango (32–8191) y no repetición entre servicios; evita colisiones que comprometan la decodificación.
 - **Selección de programa**: Cuando la fuente es TS, se detectan los canales disponibles y se ofrece su elección desde la interfaz, minimizando los errores de configuración.
 - **Controles claros**: Botones únicos para iniciar/detener reducen ambigüedades durante la operación.
- Reproducibilidad
 - **Automatización con CLI**: Todas las etapas críticas se ejecutan mediante herramientas de línea de comandos, dejando un rastro auditable de qué se ejecutó y con qué parámetros.
 - **Estructura de artefactos**: Separación de entradas de usuario, artefactos intermedios y salida final (TS), lo que permite repetir el proceso y comparar resultados.

- Limitaciones asumidas y mitigaciones

- **Entorno mixto de Python:** La generación de tablas SI/PSI utiliza scripts compatibles con Python 2.7 debido a la dependencia de *dvbobjects* [Debian, 2013]; se acota su uso solo a esa etapa y la aplicación principal permanece en Python 3, mitigando riesgos de mantenimiento.
- **Edición avanzada del TS:** No se realiza en la aplicación; se privilegia un flujo de playout/modulación confiable. Para re-multiplexación/ediciones complejas, pueden emplearse herramientas especializadas fuera del alcance de esta memoria.

3.5. Casos de uso operativos

Se describen los principales casos de uso desde la perspectiva del usuario final. Cada caso incluye propósito, precondiciones, flujo resumido y resultado/estado final.

- Previsualizar contenido

- Propósito: Revisar el material audiovisual en la GUI antes de cualquier procesamiento.
- Precondiciones: Archivo legible; la aplicación en estado *idle*.
- Flujo: Abrir archivo → cargar en reproductor embebido → reproducir/pausar, ajustar volumen y desplazarse en la barra de progreso.
- Resultado: Medio validado visualmente; estado *previsualización*. El *seek* está habilitado.

- Cargar TS existente y seleccionar el programa

- Propósito: Utilizar un Transport Stream ya construido para previsualizar o transmitir.
- Precondiciones: Archivo TS válido.
- Flujo: Abrir TS → inspección automática de programas → listado en la GUI (*Canal N*) → selección de programa.
- Resultado: TS listo para reproducción/transmisión con el programa elegido; la GUI actualiza el selector.

- Crear TS (asistente)
 - Propósito: Construir un TS final a partir de uno o dos archivos de entrada.
 - Precondiciones: Archivos de origen legibles; carpeta de salida seleccionada.
 - Flujo: (1) Fuentes → (2) Parámetros A/V y PIDs (con validación de rango y colisiones) → (3) Identificadores de red/servicio → ejecución del pipeline (preparación A/V, señalización, multiplexado CBR y estampado) con barra de progreso y registro.
 - Resultado: TS final verificado y cargado automáticamente en la GUI; selector de programa actualizado; estado *idle/previsualización*.
- Iniciar transmisión
 - Propósito: Emitir el TS final por **RF** mediante la tarjeta de modulación.
 - Precondiciones: TS cargado; parámetros de salida configurados (tarjeta, canal/frecuencia, tasa, modo).
 - Flujo: Reproducir localmente → esperar retardo controlado → iniciar modulación → bloquear *seek* durante transmisión.
 - Resultado: Señal RF activa; estado *transmisión*; la GUI muestra logs y permite *Detener*.
- Detener
 - Propósito: Finalizar ordenadamente cualquier modo de operación.
 - Precondiciones: Estado *previsualización* o *transmisión*.
 - Flujo: Invocar *Detener* → finalizar reproductor embebido → terminar proceso de transmisión → limpiar recursos.
 - Resultado: Aplicación vuelve a *idle*; recursos liberados; registro del cierre en panel de *logs*.
- Manejo de errores y validaciones
 - Propósito: Asegurar consistencia y trazabilidad ante entradas inválidas o fallas de herramientas externas.

- Precondiciones: Cualquiera.
- Flujo: Detección temprana (archivos ilegibles, valores fuera de rango, PIDs en colisión) → bloqueo de continuación → mensajes claros en la GUI y detalle en el panel de *logs*.
- Resultado: Operación segura; el usuario corrige y reintenta; se conserva traza de los eventos.

EVALUACIÓN DE ODISEASTREAM

Aquí se presenta el diseño de OdiseaStream, detallando cómo se organiza el sistema para previsualizar, construir y transmitir contenidos bajo ISDB-Tb utilizando la tarjeta DTA-115 en Linux. El capítulo expone la arquitectura por capas, el modelo de procesos y flujos de datos, los componentes y sus responsabilidades, y las decisiones de diseño que priorizan la modularidad, compatibilidad y trazabilidad del flujo completo (desde la preparación A/V y la señalización SI/PSI hasta la modulación). Asimismo, se describen los casos de uso operativos que guían la interacción del usuario con la interfaz, el reproductor embebido, el asistente de creación de TS y el control de transmisión, junto con las validaciones y registros que aseguran una operación reproducible y diagnosticable.

4.1. Interfaz gráfica

Como se puede apreciar en la figura [4.1](#), la interfaz gráfica de OdiseaStream posee una disposición limpia, funcional y centrada en facilitar el control y la supervisión de la transmisión digital. La disposición de los elementos en pantalla permite una experiencia intuitiva, evita la sobrecarga visual y reduce la posibilidad de errores operacionales.

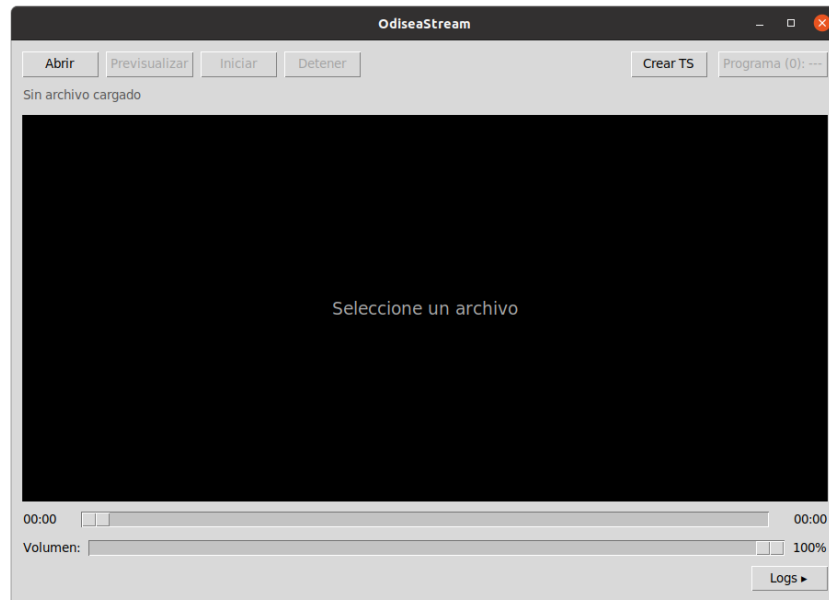


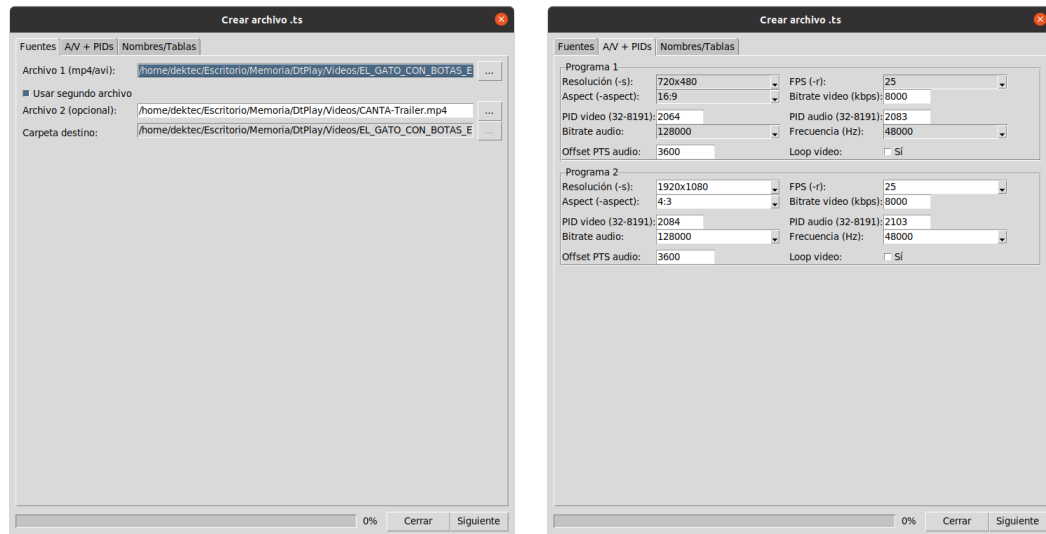
Figura 4.1: Interfaz gráfica de OdiseaStream

4.2. Generación archivo .ts

La evaluación del módulo de generación de archivos .ts dentro de OdiseaStream se centró en verificar que todas las etapas del procesamiento audiovisual se ejecutaran de manera secuencial y sin errores, desde la selección del archivo fuente hasta la obtención del archivo multiplexado y sincronizado, apto para su transmisión por la norma ISDB-Tb.

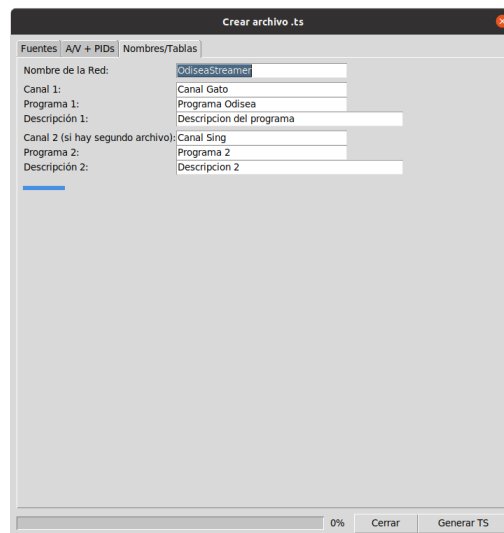
Durante las pruebas, se utilizaron archivos fuente en formato .mp4 y .avi, con diferentes resoluciones, duraciones y tasas de bits. En todos los casos, dentro de los parámetros establecidos, el sistema gestionó correctamente la conversión con los parámetros ingresados por el usuario. El ejemplo presentado será la generación con el archivo de prueba “Gato con botas: El último deseo - Tráiler” con parámetros establecidos en la figura 4.2b para la generación del archivo de 2 canales se utilizó “Canta - Tráiler” y “Gato con botas: El último deseo - Trailer”

El sistema gestionó correctamente la conversión de audio a .mp2 y de video a .m2v, mediante comandos ffmpeg. Una vez obtenidos estos archivos Elementary Stream, se eva-



(a) Selección de archivos

(b) Datos de audio/video



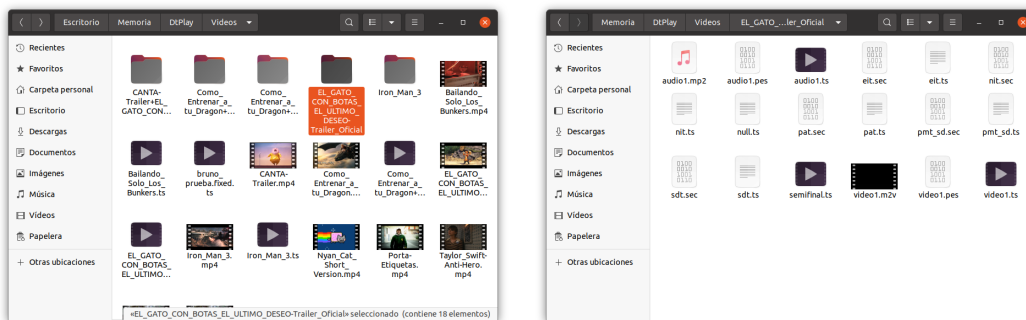
(c) Datos del canal

Figura 4.2: Creación de archivo .ts

luó la correcta ejecución de las herramientas `esvideo2pes` y `esaudio2pes`, verificando que los archivos `.pes` de audio y video generados fueron compatibles y se transformaron mediante `pesvideo2ts` y `pesaudio2ts` en flujos de transporte individuales `video.ts` y `audio.ts`.

Posteriormente, el sistema ejecutó correctamente el script para la generación de tablas SI/PSI necesarias para la transmisión digital. Esto incluye las tablas PAT, PMT, SDT y NIT, las cuales fueron empaquetadas en formato `.ts` y luego multiplexadas junto con el video, el audio y el archivo `null.ts` mediante `tscbrmuxer`; finalmente, se aplicó la herramienta `tsstamp` para garantizar la correcta sincronización de los paquetes. El programa genera carpetas auxiliares para almacenar los archivos necesarios para la generación del archivo `.ts` final. Para el caso de 1 canal, se puede observar la carpeta creada y su contenido en la figura 4.3, mientras que para el caso de 2 canales, la carpeta auxiliar creada y su contenido se observan en la figura 4.4

El proceso concluye correctamente y genera un archivo `.ts` compatible con transmisión bajo la norma ISDB-Tb utilizando la tarjeta moduladora DTA-115, durante el proceso de creación de archivos `.ts`.

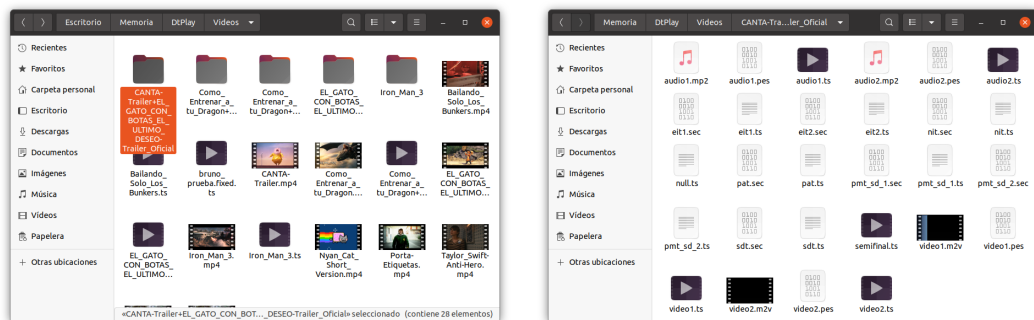


(a) Carpeta auxiliar creada

(b) Contenido carpeta auxiliar

Figura 4.3: Carpeta auxiliar 1 canal

Es importante mencionar también que OdiseaStream cuenta con control de errores, en caso de que se intente generar un archivo `.ts` sin un archivo fuente, como se aprecia en la figura 4.5a y en la figura 4.5b. Asimismo, se presenta un error si se intenta generar un archivo

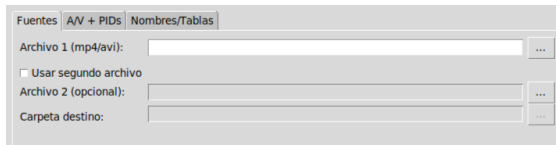


(a) Carpeta auxiliar creada 2 canales

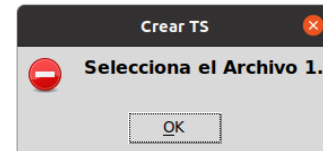
(b) Contenido carpeta auxiliar 2 canales

Figura 4.4: Carpeta auxiliar 2 canales

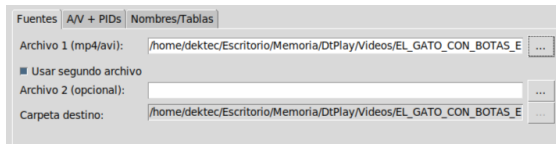
.ts con dos fuentes audiovisuales pero sin seleccionar la segunda, como se aprecia en la figura 4.5c y en la figura 4.5d. Para asegurar que los PIDs elegidos se encuentren dentro de los valores esperados, figura 4.5e y figura 4.5f. Y, finalmente, para asegurar que todos los PIDs de audio y video sean diferentes entre sí, como se aprecia en la figura 4.5g y en la figura 4.5h.



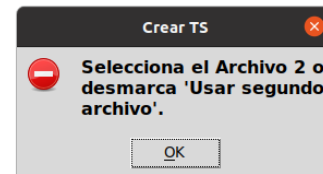
(a) Primer archivo no seleccionado



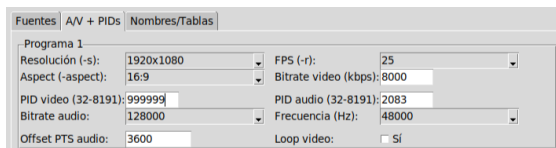
(b) Error al no seleccionar el primer archivo



(c) Segundo archivo no seleccionado



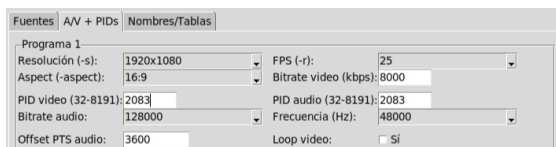
(d) Error al no seleccionar segundo archivo



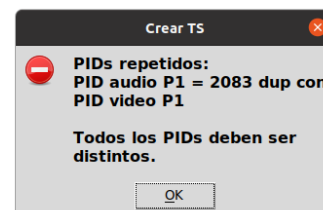
(e) PID fuera de rango



(f) Error al tener PID fuera de rango



(g) Repetir PID



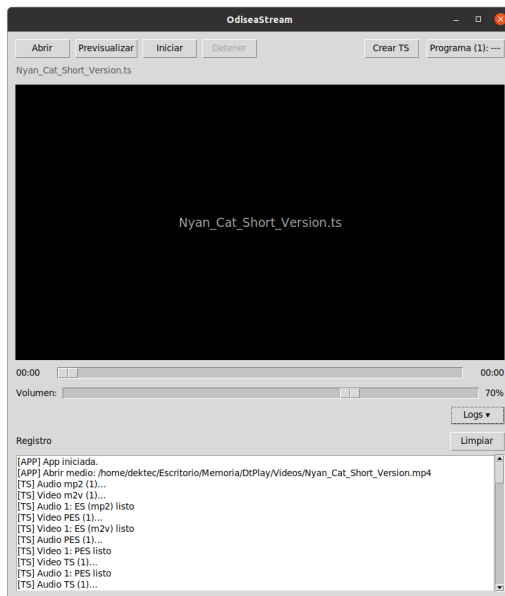
(h) Error al repetir PID

Figura 4.5: Manejo de errores

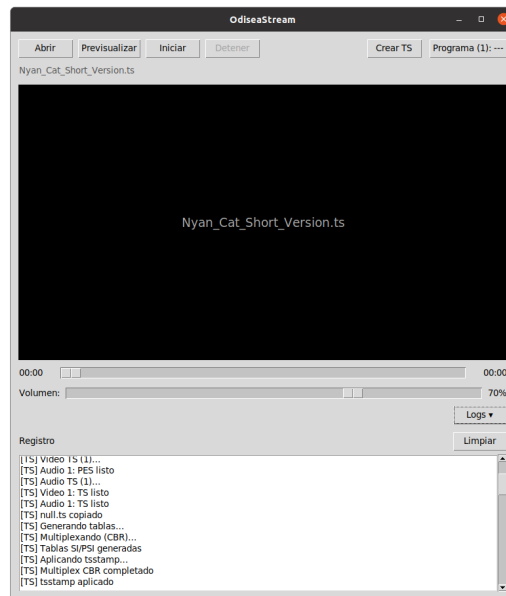
OdiseaStream imprime todo el proceso en la pestaña de logs de forma resumida como se

ve en la figura [4.6](#) y para usuarios más avanzados lo imprime en consola de comandos lo que permite la trazabilidad y reproducibilidad del proceso completo, como se puede apreciar en la figura [4.7](#)

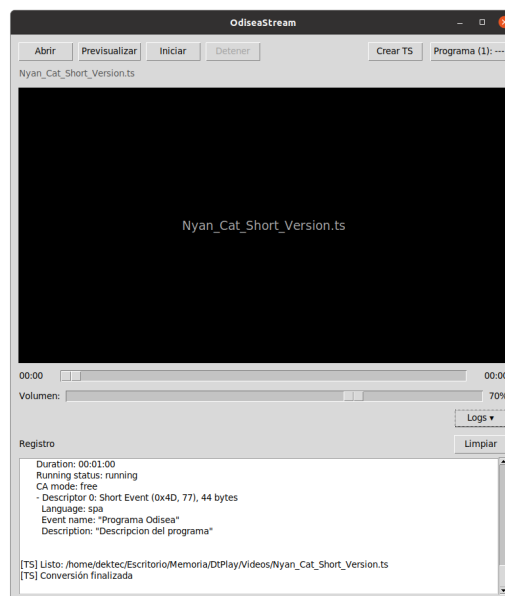
Finalmente, para verificar la estructura del archivo .ts generado y la presencia de tablas PSI/SI se hace uso de herramientas de tsduck, explícitamente tstable, como se aprecia en la figura [4.8](#)



(a) Primera parte del proceso

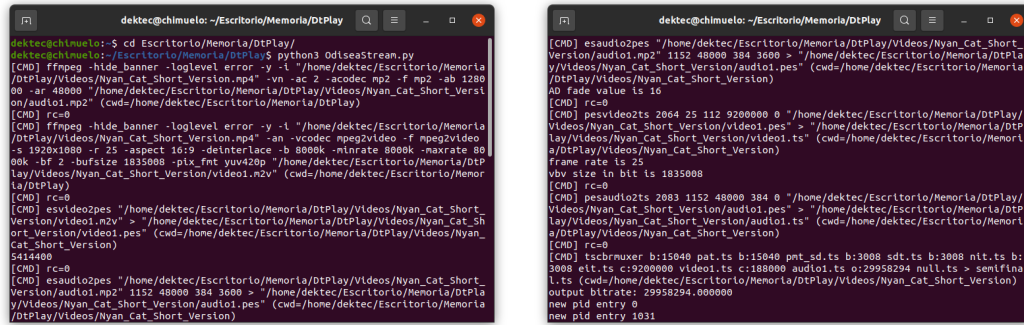


(b) Segunda parte del proceso



(c) Tercera parte del proceso

Figura 4.6: Proceso en ventana de logs



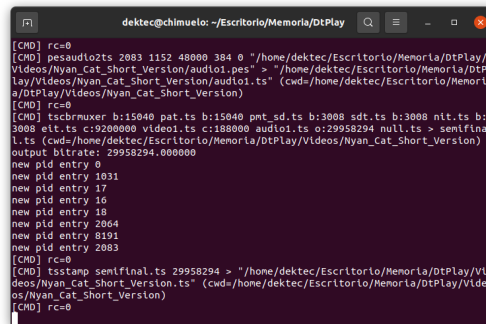
```

dektec@chimuelo: ~/Escritorio/Memoria/DtPlay
[CMD] dektec@chimuelo:~/Escritorio/Memoria/DtPlay$ python3 odiseaStream.py
[CMD] ffmpeg -hide_banner -loglevel error -y -i "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version.mp4" -vn -ac 2 -acodec mp2 -f mp2 -ab 128000 -ar 48000 "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol.mp2" (cwd=/home/dektec/Escritorio/Memoria/DtPlay)
[CMD] rc=0
[CMD] ffmpeg -hide_banner -loglevel error -y -i "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version.mp4" -an -vcodec mpeg2video -f mpeg2video -s 1920x1080 -r 25 -aspect 16:9 -deinterlace -b 8000k -minrate 8000k -maxrate 8000k -bf 2 -bufsize 1835008 -pix_fmt yuv420p "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/video1.n2v" (cwd=/home/dektec/Escritorio/Memoria/DtPlay)
[CMD] rc=0
[CMD] esvideo2ts "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/video1.n2v" > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/video1.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0
[CMD] esaudio2ts "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol.mp2" 1152 48000 384 3600 > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0
[CMD] pesaudio2ts 2083 1152 48000 384 0 "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol.ts" > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol1.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0
[CMD] tscbrmuxer b:15040 pat.ts b:15040 pmt.sd.ts b:3008 sdt.ts b:3008 nit.ts b:3008 elt.ts c:92000000 video1.ts c:188000 audiol.ts o:29958294 null.ts > semifinal1.ts (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
output bitrate: 29958294.000000
new pid entry 0
new pid entry 1031
new pid entry 17
new pid entry 16
new pid entry 18
new pid entry 2064
new pid entry 8191
new pid entry 2083
[CMD] rc=0
[CMD] esstamp semifinal1.ts 29958294 > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0

```

(a) Primera parte del proceso

(b) Segunda parte del proceso



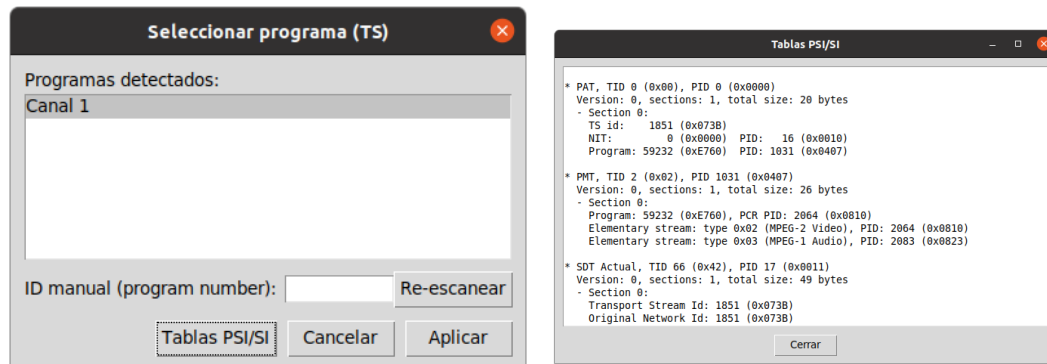
```

dektec@chimuelo: ~/Escritorio/Memoria/DtPlay
[CMD] rc=0
[CMD] pesaudio2ts 2083 1152 48000 384 0 "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol.ts" > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version/audiol1.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0
[CMD] tscbrmuxer b:15040 pat.ts b:15040 pmt.sd.ts b:3008 sdt.ts b:3008 nit.ts b:3008 elt.ts c:92000000 video1.ts c:188000 audiol.ts o:29958294 null.ts > semifinal1.ts (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
output bitrate: 29958294.000000
new pid entry 0
new pid entry 1031
new pid entry 17
new pid entry 16
new pid entry 18
new pid entry 2064
new pid entry 8191
new pid entry 2083
[CMD] rc=0
[CMD] esstamp semifinal1.ts 29958294 > "/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version.ts" (cwd=/home/dektec/Escritorio/Memoria/DtPlay/Videos/Nyan_Cat_Short_Version)
[CMD] rc=0

```

(c) Tercera parte del proceso

Figura 4.7: Proceso en consola



(a) Botón "Programa"

(b) Tablas PSI/SI

Figura 4.8

4.3. Previsualización local

La funcionalidad de previsualización local de OdiseaStream fue evaluada con el objetivo de comprobar su correcto desempeño al reproducir el contenido del archivo .ts dentro de la propia interfaz gráfica, sin requerir software externo. Esta característica es fundamental, ya que permite al usuario validar visualmente que el archivo seleccionado sea el correcto antes de ser transmitido. Se hace uso del reproductor VLC mediante python-vlc para la previsualización del contenido mediante el botón "Previsualizar"

La carga de archivos se puede realizar directamente desde un archivo .ts previamente existente, como se puede apreciar en la figura 4.9 para esta prueba se utilizó un archivo previamente existente con 1 canal.

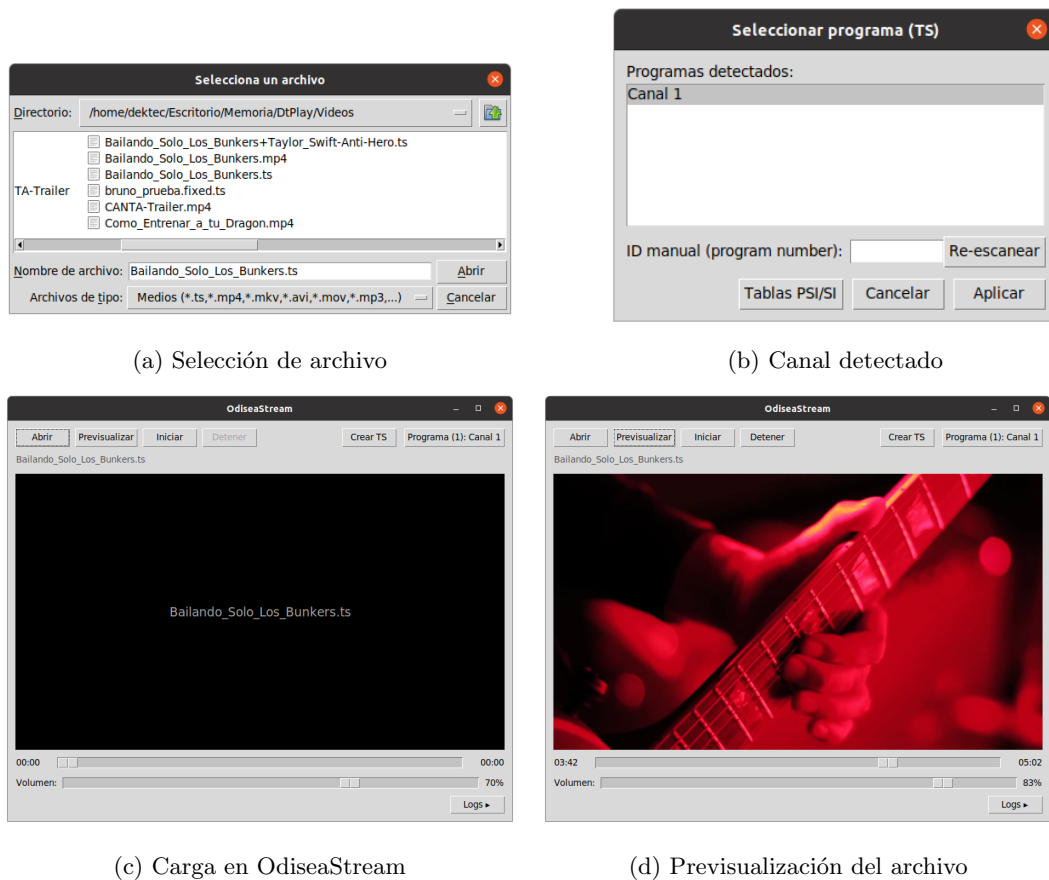
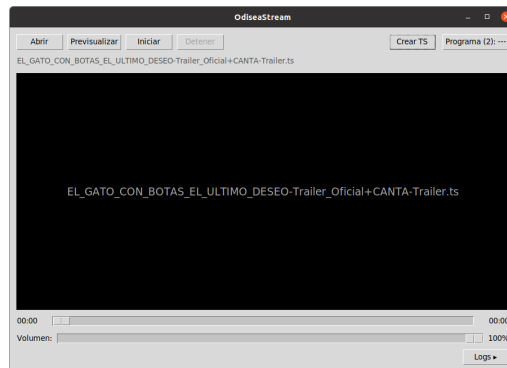
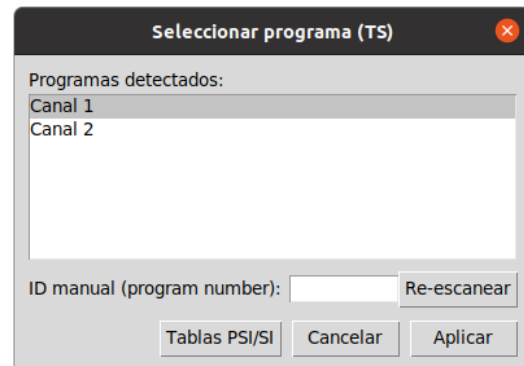


Figura 4.9: Previsualización desde archivo previamente existente

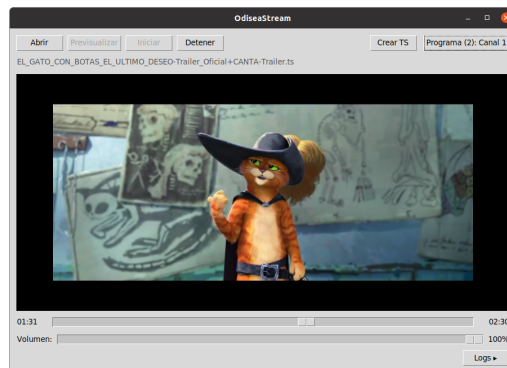
También se puede hacer uso de un archivo .ts generado por el mismo programa, como es el caso del archivo generado en el capítulo 4.2. El archivo generado en dicho capítulo posee dos canales; para estas situaciones OdiseaStream posee un botón de "Programa" para seleccionar cuál fuente de video previsualizar, como se aprecia en la figura 4.10b. La selección de alguno de estos canales lleva a su correcta previsualización, como se ve en las figuras 4.10c y 4.10d



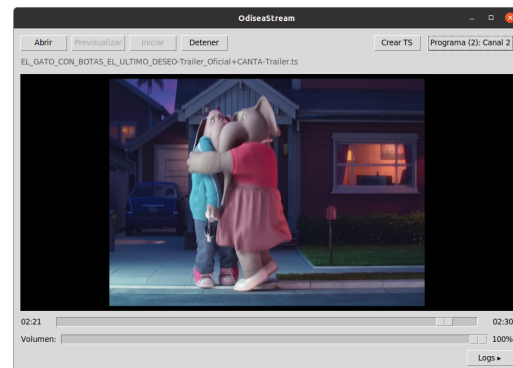
(a) Selección de archivo .ts con 2 canales



(b) Selector de canal



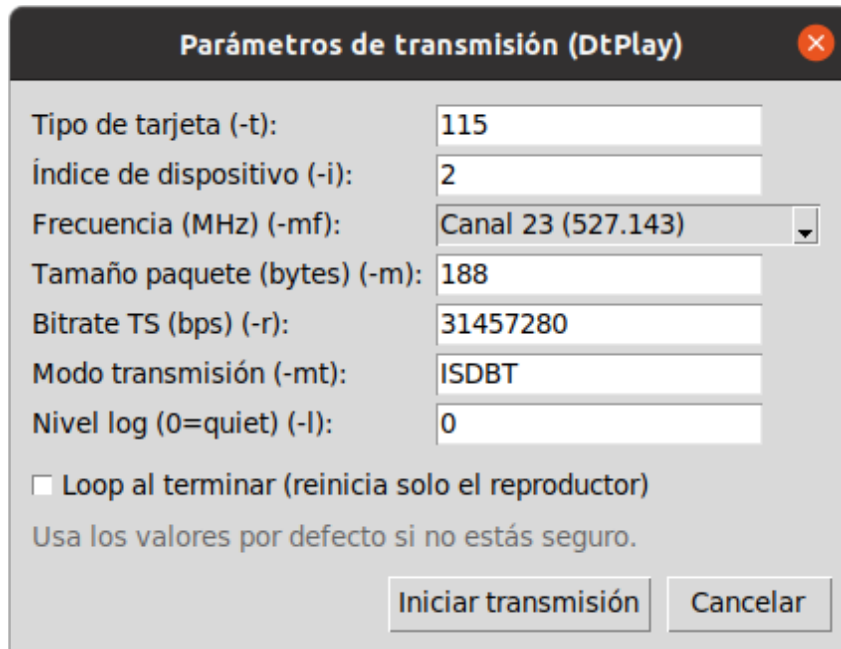
(c) Previsualización canal 1



(d) Previsualización canal 2

Figura 4.10: Previsualización de archivo generado por OdiseaStream

4.4. Transmisión por tarjeta moduladora DTA-115



Parámetros de transmisión (DtPlay)

Tipo de tarjeta (-t): 115

Índice de dispositivo (-i): 2

Frecuencia (MHz) (-mf): Canal 23 (527.143)

Tamaño paquete (bytes) (-m): 188

Bitrate TS (bps) (-r): 31457280

Modo transmisión (-mt): ISDBT

Nivel log (0=quiet) (-l): 0

☐ Loop al terminar (reinicia solo el reproductor)

Usa los valores por defecto si no estás seguro.

Iniciar transmisión Cancelar

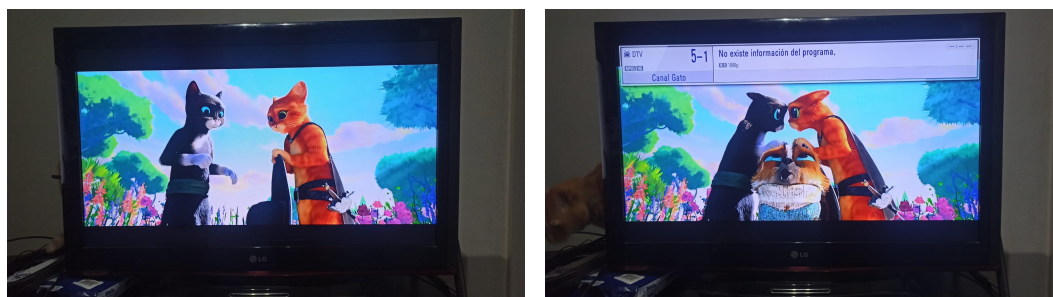
Figura 4.11: Selección de parámetros de transmisión

La etapa final de evaluación de OdiseaStream consiste en la transmisión efectiva del archivo .ts generado, utilizando la tarjeta moduladora DTA-115. Esta tarea se realiza mediante la ejecución del software DtPlay al presionar el botón “Iniciar”, el cual transmite el flujo en tiempo real utilizando los parámetros previamente definidos por el usuario en la interfaz del sistema, como se observa en la figura [4.11](#)

Para validar esta funcionalidad, se realizaron pruebas tanto con transmisiones de un canal como con transmisiones de dos canales. En ambos casos, se evaluó el reconocimiento del canal en televisores compatibles con la norma ISDB-Tb, así como la fidelidad del contenido audiovisual y la integridad de las tablas SI/PSI.

4.4.1. Un canal

En esta prueba se utilizó el archivo .ts generado con un solo servicio de televisión digital, correspondiente al tráiler de la película “El Gato con Botas: El último deseo”. Una vez iniciado el proceso de transmisión desde OdiseaStream, el receptor de televisión sintonizó correctamente el canal, mostrando tanto el contenido audiovisual como el nombre del canal y el formato; como se puede apreciar en la figura 4.12. Se puede apreciar en la figura 4.12a la visualización del contenido en pantalla, que contiene las escenas del tráiler mencionado anteriormente. En la figura 4.12b se puede apreciar la interfaz de la televisión, junto con el nombre del canal y el formato del contenido.



(a) Recepción del televisor

(b) Datos de la transmisión

Figura 4.12: Transmisión 1 canal

Finalmente, en la figura 4.13 se observa el entorno completo de las pruebas de transmisión, donde se visualiza simultáneamente el archivo en ejecución en el televisor y la ejecución de OdiseaStream en la computadora con Ubuntu; se observa que el previsualizador está ligeramente adelantado a la transmisión, debido al tiempo de codificación/decodificación que se produce en el televisor.

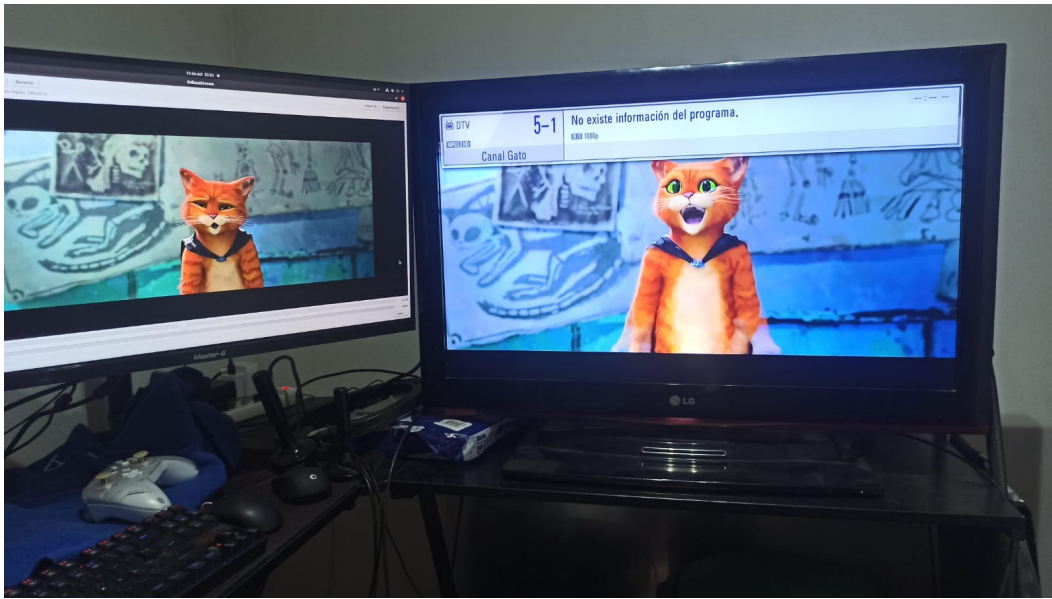
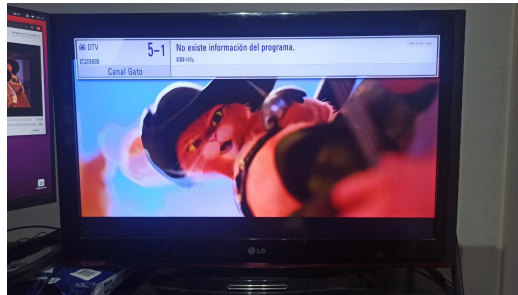


Figura 4.13: Transmisión en vivo

4.4.2. Dos canales

OdiseaStream también fue evaluado en su capacidad para transmitir dos servicios audiovisuales de televisión de forma simultánea dentro del mismo flujo TS. Para ello, se seleccionaron dos archivos de video distintos, correspondientes a los tráilers de las películas “Canta” y “El Gato con Botas: El último deseo” y se generó un archivo .ts con ambos canales, cada uno con sus correspondientes tablas SI/PSI gracias al script `generar_tablas_dual.py`, incluido en OdiseaStream.

La figura 4.14 muestra el resultado de esta transmisión múltiple. El receptor de televisión fue capaz de reconocer dos canales distintos dentro del mismo flujo, en la figura 4.14a se aprecia la correcta transmisión de “El Gato con Botas: El último deseo”, mientras que en la figura 4.14b se observa la correcta recepción del canal que contiene el trailer de “Canta”.



(a) Recepción del canal 1



(b) Recepción del canal 2

Figura 4.14: Transmisión de 2 canales

4.5. Matriz de trazabilidad

La siguiente matriz vincula los requisitos funcionales del sistema con los componentes que los satisfacen y con la evidencia incluida en este documento (figuras, secciones y capítulo de evaluación).

ID	Requisito	Componentes implicados	Evidencia
R1	Previsualizar contenido dentro de la aplicación.	GUI; Reproductor embebido; Gestor de medios.	§ Capítulo 4.3 § Figura 4.9d § Figura 4.10c § Figura 4.10d
R2	Cargar un TS existente y seleccionar el programa a emitir.	Gestor de medios (inspección TS); GUI (selector); Validador de identificadores.	§ Figura 4.9a § Figura 4.9c § Figura 4.10b
R3	Construir un TS final a partir de 1 o 2 fuentes con parámetros configurables.	Motor de preparación A/V; Generador de señalización (SI/PSI); Constructor de TS.	§ Capítulo 4.2 § Figura 4.2
R4	Manejo de errores por parte de OdiseaStream	Validador de identificadores; GUI (mensajes y bloqueo).	§ Figura 4.5
R5	Verificar estructura del servicio y presencia de tablas PSI/SI en el TS final.	Constructor de TS; (herramienta de inspección TS); Registro y diagnóstico.	§ Figura 4.8
R6	Transmitir por RF mediante tarjeta DTA-115 con parámetros configurables.	Control de transmisión (hardware); GUI; Reproductor embebido (sincronización).	§ Capítulo 4.4 § Figura 4.11 § Figura 4.12 § Figura 4.13 § Figura 4.14
R7	Mantener trazabilidad y reproducibilidad del proceso.	Registro y diagnóstico; Orquestación; Estructura de artefactos.	§ Figura 4.7

Tabla 4.1: Matriz de trazabilidad entre requisitos, componentes y evidencia documental.

CONCLUSIONES

El desarrollo de OdiseaStream permitió la implementación de un sistema integrado, funcional y autónomo para la transmisión de contenido audiovisual utilizando la tarjeta moduladora DTA-115 de Dektec, bajo la norma ISDB-Tb. A lo largo de este proyecto, se logró integrar exitosamente diversas herramientas de conversión, empaquetado, generación de tablas SI/PSI, sincronización y transmisión, todo a través de una interfaz gráfica diseñada desde cero.

El sistema demostró un comportamiento estable y confiable durante las pruebas de generación de archivos .ts, así como en la transmisión de un canal y en la transmisión de dos canales de manera simultánea. La inclusión de funciones de previsualización local, creación de archivos .ts y control de transmisión permitió consolidar una solución integral y operativa, que no requiere intervención del usuario en la línea de comandos. Todo el proceso se realiza directamente a través de OdiseaStream.

Entre los principales logros del proyecto, destacan:

- La automatización completa del flujo de transformación de archivos de video a .ts conforme al estándar ISDB-Tb.
- El diseño de una interfaz gráfica clara, funcional y amigable permite operar todo el proceso desde un entorno visual.

- La integración de la tarjeta moduladora DTA-115 para la transmisión directa ha dado resultados exitosos en pruebas reales con receptores de televisión digital.
- La posibilidad de incluir más de un canal dentro del mismo flujo digital permite escenarios de transmisión avanzada.

En términos generales, OdiseaStream cumple con gran parte de los objetivos propuestos al inicio del proyecto, brindando una solución robusta, adaptable y de utilidad práctica para el entorno de enseñanza, experimentación o implementación de transmisores de baja potencia basados en software libre. Sin embargo, ocurrieron modificaciones respecto al planteamiento original; se hizo un cambio en el lenguaje de programación utilizado; la propuesta inicial fue C/C++ debido a su bajo costo computacional y alto rendimiento, sin embargo, se realizó un cambio al uso de Python gracias al repertorio de herramientas que se utilizaron para el desarrollo del proyecto. Además, no se alcanzó el objetivo de transmitir un flujo de video proveniente de otra máquina, entendido como transmisiones en vivo. Esto se debe a que el **SDK** ofrecido por el fabricante requiere archivos .ts íntegros para su transmisión exitosa.

5.1. Trabajo futuro

Si bien OdiseaStream cumple con su propósito, existen diversas líneas de desarrollo que pueden ser abordadas para ampliar sus funcionalidades y mejorar su desempeño:

- Soporte para transmisión en vivo: Actualmente, el sistema trabaja sobre archivos locales, integrar una fuente de captura en vivo permitiría usar el sistema en contextos de señal en vivo.
- Compatibilidad con otras tarjetas moduladoras: Ampliar la compatibilidad a otras marcas o modelos fuera del fabricante de la tarjeta utilizada.

BIBLIOGRAFÍA

[Association of Radio Industries and Businesses (ARIB), 2014] Association of Radio Industries and Businesses (ARIB) (2014). Transmission system for digital terrestrial television broadcasting: Arib std-b31, version 2.2-e1. Disponible en: https://www.arib.or.jp/english/html/overview/doc/6-STD-B31v2_2-E1.pdf, consultado en octubre de 2025.

[Avalpa Digital Engineering, 2025] Avalpa Digital Engineering (2025). Opencaster: tools to play out and multiplex mpeg transport streams. <https://github.com/aventuri/opencaster>. Consultado en octubre de 2025.

[Consejo Nacional de Televisión de Chile, 2011] Consejo Nacional de Televisión de Chile (2011). ¿qué es tvd? Disponible en: <https://www.tvd.cl/que-es-tvd/>, consultado en julio de 2025.

[Debian, 2013] Debian (2013). Opencaster: empaquetado y librería dvbobjects. <https://sources.debian.org/patches/opencaster/3.2.2%2Bdfsg-1/python.patch/>. Consultado en octubre de 2025.

[DekTec Digital Video BV, 2011] DekTec Digital Video BV (2011). Dta-115 leaflet: Multi-standard modulator with vhf/uhf upconverter. Disponible en: <https://www.dektec.com/products/pci/dta-115/downloads/DTA-115%20Leaflet.pdf>, consultado en julio de 2025.

[DekTec Digital Video BV, 2024] DekTec Digital Video BV (2024). Downloads – drivers and sdks (linuxsdk/dtapi). <https://www.dektec.com/downloads/SDK/>. Paquete LinuxSDK para DTA/DTU/DTE; versión utilizada: 2024.04.0. Consultado en octubre de 2025.

[DekTec Digital Video BV, 2025] DekTec Digital Video BV (2025). Streamxpress – versatile stream player for r&d, testing and demos. Disponible en: <https://www.dektec.com/products/applications/StreamXpress/>, consultado en julio de 2025.

[Digital Broadcasting Experts Group (DiBEG), 2007] Digital Broadcasting Experts Group (DiBEG) (2007). Transmission system of isdb-t (seminar slides). Disponible en: https://www.dibeg.org/wp/wp-content/uploads/news/0706_3Argentina_ISDB-T_seminar/Argentina_ISDB-T_seminar_No3_transmission_system.pdf, consultado en octubre de 2025.

[FFmpeg Project, 2025a] FFmpeg Project (2025a). About ffmpeg. <https://www.ffmpeg.org/about.html>. Consultado en octubre de 2025.

[FFmpeg Project, 2025b] FFmpeg Project (2025b). ffmpeg documentation. <https://ffmpeg.org/ffmpeg.html>. Consultado en octubre de 2025.

[Gobierno de Chile, 2014] Gobierno de Chile (2014). Ley n° 20.749: Permite la introducción de la televisión digital terrestre en Chile. <https://www.bcn.cl/leychile/navegar?idNorma=1060307>. Ley publicada en el Diario Oficial el 29 de mayo de 2014.

[Moncayo Flores, 2014] Moncayo Flores, T. J. (2014). Diseño e implementación de un transmisor de pruebas de tv digital terrestre isdb-tb. Master's thesis, Escuela Politécnica Nacional. Cita el material de LIFIA: “OpenCaster para SATVD-T”. Consultado en octubre de 2025.

[Python Software Foundation, 2020] Python Software Foundation (2020). Python 2.7 documentation. <https://docs.python.org/2.7/>. Última versión 2.7.18 (abril de 2020). Consultado en octubre de 2025.

[Python Software Foundation, 2025] Python Software Foundation (2025). tkinter — python interface to tcl/tk. <https://docs.python.org/3/library/tkinter.html>. Consultado en octubre de 2025.

[Subsecretaría de Telecomunicaciones, 2009] Subsecretaría de Telecomunicaciones (2009). Gobierno de Chile adopta norma de televisión digital para el país. Disponible en: <https://www.subtel.gob.cl/>

DEPARTAMENTO DE ELECTRÓNICA

[gobierno-de-chile-adopta-norma-de-television-digital-para-el-pais/](#), consultado en octubre de 2025.

[TSDuck Project, 2025a] TSDuck Project (2025a). Tsdock – the mpeg transport stream toolkit. <https://tsduck.io/>. Consultado en octubre de 2025.

[TSDuck Project, 2025b] TSDuck Project (2025b). tsduck/tsduck: Mpeg transport stream toolkit. <https://github.com/tsduck/tsduck>. Consultado en octubre de 2025.

[Universidad Técnica Federico Santa María, 2022] Universidad Técnica Federico Santa María (2022). Televisión digital bajo norma isdb-tb: Guía de práctica con dta-115. https://investigacion.electronica.usm.cl/~tvd/docs/guias/Informe_Practica_DTA115.pdf. Incluye instrucciones de instalación: OpenCaster 2.4 + Patch LIFIA (rev362). Consultado en octubre de 2025.

[VideoLAN, 2025] VideoLAN (2025). Python ctypes-based bindings for libvlc (python-vlc). <https://python-vlc.readthedocs.io/>. Consultado en octubre de 2025.

[VideoLAN / Olivier Aubert, 2025] VideoLAN / Olivier Aubert (2025). python-vlc (ctypes bindings for libvlc). <https://pypi.org/project/python-vlc/3.0.18122/>. Consultado en octubre de 2025.

ANEXOS

Archivo “OdiseaStream.py”

```
1  import os
2  import sys
3  import re
4  import threading
5  import subprocess
6  import queue
7  from typing import Optional, List, Set, Tuple, Dict
8
9  # --- Linux: inicializar X11 para hilos y preferir XCB sobre Xlib ---
10 if sys.platform.startswith("linux"):
11     try:
12         import ctypes
13         ctypes.CDLL("libX11.so.6").XInitThreads()
14     except Exception as e:
15         print("Aviso: no se pudo llamar XInitThreads():", e)
16     os.environ.setdefault("LIBVLC_USE_XLIB", "0")
17
18 import tkinter as tk
19 from tkinter import ttk, filedialog, messagebox
20
21 try:
22     import vlc # pip install python-vlc
23 except Exception as e:
24     print("No se pudo importar 'python-vlc'. Instala con: pip install python-vlc")
25     raise
```

```

26
27
28 # =====
29 # Utilidades
30 # =====
31 def _fmt_ms(ms: int) -> str:
32     s = max(0, int(ms) // 1000)
33     h, rem = divmod(s, 3600)
34     m, s = divmod(rem, 60)
35     return f"{h:02d}:{m:02d}:{s:02d}" if h else f"{m:02d}:{s:02d}"
36
37
38 def run_cmd(cmd: List[str], cwd: Optional[str] = None) ->
↳ subprocess.CompletedProcess:
39     """Comando con captura (se usa para 'tstables' donde necesitamos stdout)."""
40     return subprocess.run(
41         cmd, cwd=cwd, text=True, capture_output=True, errors="replace", shell=False
42     )
43
44
45 def run_cmd_shell_quiet(cmd: str, cwd: Optional[str] = None) -> int:
46     """Antes: silenciaba stdout/stderr.
47     Ahora: RESTAURA la salida a consola (para ver si 'Crear TS' progresa).
48     Ejecuta el comando en shell y deja que imprima en la consola del proceso.
49     """
50     try:
51         # Imprimir el comando para trazabilidad
52         print(f"[CMD] {cmd} (cwd={cwd or os.getcwd()})")
53         r = subprocess.run(
54             cmd,
55             cwd=cwd,
56             text=False,          # binario: respetar streams de herramientas
↳     externas
57             stdout=None,        # None = hereda stdout del proceso padre (visible
↳     en consola)

```

```

58         stderr=None,          # None = hereda stderr (visible en consola)
59         shell=True,
60         check=False,
61     )
62     print(f"[CMD] rc={r.returncode}")
63     return r.returncode
64 except Exception as e:
65     print(f"[CMD] ERROR: {e}")
66     return 1
67
68
69
70 # =====
71 # App principal con VLC
72 # =====
73 class VLCApp:
74     LOGS_PADY = 12 # espacio superior del panel de logs (debe coincidir con pack
    ↪ pady)
75
76     def __init__(self, root):
77         self.root = root
78         self.root.title("OdiseaStream")
79
80         # Registrar el hilo UI
81         self._ui_thread = threading.current_thread()
82
83         # Tamaño base (sin logs)
84         BASE_W, BASE_H = 900, 610 # altura base 610 para que "Logs" no se corte
85         self.root.geometry(f"{BASE_W}x{BASE_H}")
86         self.base_min_w, self.base_min_h = 760, 520
87         self.root.minsize(self.base_min_w, self.base_min_h)
88
89         # --- Config y nivel de logs ---
90         self.app_dir = os.path.abspath(os.path.dirname(__file__))
91         self.log_path = os.path.join(self.app_dir, "vlc.log")

```

```

92     self.log_file_pos = 0
93     self.log_job = None
94     self.logs_visible = False
95     self.size_before_logs = None
96     VERBOSE = int(os.getenv("VLC_VERBOSE", "2")) # 0=err,1=warn,2=info,3=debug
97
98     vlc_args = [f"--verbose={VERBOSE}",
99                 f"--file-logging", f"--logfile={self.log_path}",
100                 ↪ f"--log-verbose={VERBOSE}"]
101
102     if sys.platform.startswith("linux"):
103         vlc_args += ["--no-xlib", "--vout=xcb"]
104
105     self.instance = vlc.Instance(*vlc_args)
106     self.player = self.instance.media_player_new()
107     self.events = self.player.event_manager()
108
109     # Estado
110     self.current_media_path: Optional[str] = None
111     self.duration_ms = 0
112     self.update_job = None
113     self.placeholder_visible = False
114
115     # TS/Program
116     self.current_program_id: Optional[int] = None # program_number
117     # Lista de programas detectados: dicts con {'program': int, 'upid':
118     ↪ Optional[int]}
119     self.program_list: List[Dict[str, Optional[int]]] = []
120
121     # DtPlay
122     self.tx_process: Optional[subprocess.Popen] = None
123     self.tx_loop_enabled: bool = False # << LOOP (UI) para coordinar SOLO el
124     ↪ reproductor embebido
125
126     # Flags previsualización/seek
127     self.is_preview = False

```

```

124     self.dragging_seek = False
125
126     # Modo para el botón Detener: "idle" / "preview" / "tx"
127     self.mode: str = "idle"
128
129     # ===== Sistema de LOGS con cola y drenaje =====
130     self.log_queue: "queue.Queue[str]" = queue.Queue()
131     self._log_drain_job: Optional[str] = None
132     self.MAX_LOG_CHARS = 300_000 # cota del Text
133     self._current_log_chars = 0
134
135     # ---- UI ----
136     outer = ttk.Frame(root, padding=12)
137     outer.pack(fill="both", expand=True)
138
139     # Área principal (todo menos logs)
140     self.main_area = ttk.Frame(outer)
141     self.main_area.pack(fill="both", expand=True)
142
143     # Barra superior
144     topbar = ttk.Frame(self.main_area)
145     topbar.pack(fill="x", pady=(0, 10))
146
147     self.btn_abrir = ttk.Button(topbar, text="Abrir",
148     ↪ command=self.abrir_archivo)
149
150     self.btn_abrir.pack(side="left", padx=(0, 8))
151
152     # Previsualizar
153     self.btn_inicio = ttk.Button(topbar, text="Previsualizar",
154     ↪ command=self.reproducir_desde_inicio,
155     ↪ state="disabled")
156
157     self.btn_inicio.pack(side="left", padx=(0, 8))
158
159     # Iniciar (TX)

```

```

156         self.btn_tx_start = ttk.Button(topbar, text="Iniciar",
    ↪     command=self.tx_iniciar, state="disabled")
157     self.btn_tx_start.pack(side="left", padx=(0, 8))
158
159     # Botón Detener (COMBINADO)
160     self.btn_detener = ttk.Button(topbar, text="Detener",
161                                   command=self.detener, state="disabled")
162     self.btn_detener.pack(side="left", padx=(0, 8))
163
164     # A la DERECHA: Programa y Crear TS
165     self.btn_prog = ttk.Button(topbar, text="Programa (0): ---",
    ↪     command=self.abrir_selector_programa, state="disabled")
166     self.btn_prog.pack(side="right", padx=(0, 0))
167     self.btn_crear_ts = ttk.Button(topbar, text="Crear TS",
    ↪     command=self.crear_ts_dialog)
168     self.btn_crear_ts.pack(side="right", padx=(0, 12))
169
170     # --- Estado (archivo): debajo de los botones y sobre el reproductor ---
171     estado = ttk.Frame(self.main_area)
172     estado.pack(fill="x", pady=(0, 6))
173     self.lbl_archivo = ttk.Label(estado, text="Sin archivo cargado",
    ↪     foreground="#555")
174     self.lbl_archivo.pack(side="left")
175
176     # Visor de video embebido
177     self.video_frame = tk.Frame(self.main_area, bg="#000000", height=420)
178     self.video_frame.pack(fill="both", expand=True, pady=(6, 10))
179     self.placeholder = tk.Label(self.video_frame, text="Selecione un archivo",
180                                fg="AAAAAA", bg="#000000",
    ↪     font=("TkDefaultFont", 14))
181
182     self._show_placeholder()
183     self._bind_player_to_widget()
184
185     # Progreso + tiempo
186     prog = ttk.Frame(self.main_area)

```

```

186     prog.pack(fill="x", pady=(0, 6))
187     self.lbl_time = ttk.Label(prog, text="00:00", width=7, anchor="w")
188     self.lbl_time.pack(side="left")
189
190     self.progress_var = tk.DoubleVar(value=0.0)
191     self.progress = ttk.Scale(prog, from_=0, to=1000, orient="horizontal",
192                               variable=self.progress_var)
193     self.progress.pack(side="left", fill="x", expand=True, padx=6)
194     self.progress.bind("<Button-1>", self._on_seek_start, add=False)
195     self.progress.bind("<Bi-Motion>", self._on_seek_drag, add=False)
196     self.progress.bind("<ButtonRelease-1>", self._on_seek_release, add=False)
197
198     self.lbl_dur = ttk.Label(prog, text="00:00", width=7, anchor="e")
199     self.lbl_dur.pack(side="right")
200
201     # Volumen
202     vol_frame = ttk.Frame(self.main_area)
203     vol_frame.pack(fill="x", pady=(6, 0))
204     ttk.Label(vol_frame, text="Volumen:").pack(side="left", padx=(0, 8))
205     self.volumen = tk.IntVar(value=70)
206     self.lbl_vol = ttk.Label(vol_frame, text=f"{self.volumen.get()}%")
207     self.lbl_vol.pack(side="right", padx=(8, 0))
208     self.scale_vol = ttk.Scale(vol_frame, from_=0, to=100, orient="horizontal")
209     self.scale_vol.pack(side="left", fill="x", expand=True)
210     self.scale_vol.set(self.volumen.get())
211     self.scale_vol.configure(command=self._on_vol_change)
212
213     # Barra inferior con botón Logs
214     botbar = ttk.Frame(self.main_area)
215     botbar.pack(fill="x", pady=(10, 0))
216     self.btn_logs = ttk.Button(botbar, text="Logs ", command=self.toggle_logs)
217     self.btn_logs.pack(side="right")
218
219     # Panel de logs (hermano de main_area)
220     self.log_frame = ttk.Frame(outer)

```

```

221         top_logs = ttk.Frame(self.log_frame)
222         top_logs.pack(fill="x")
223         ttk.Label(top_logs, text="Registro (libVLC + app +
↳ herramientas)").pack(side="left")
224         self.btn_clear = ttk.Button(top_logs, text="Limpiar",
↳ command=self.clear_logs)
225         self.btn_clear.pack(side="right")
226         body_logs = ttk.Frame(self.log_frame)
227         body_logs.pack(fill="both", expand=True, pady=(6, 0))
228         self.log_text = tk.Text(body_logs, height=12, wrap="none",
229                                 state="disabled", font=("TkFixedFont", 10))
230         yscroll = ttk.Scrollbar(body_logs, orient="vertical",
↳ command=self.log_text.yview)
231         self.log_text.configure(yscrollcommand=yscroll.set)
232         self.log_text.pack(side="left", fill="both", expand=True)
233         yscroll.pack(side="right", fill="y")
234
235         # Volumen inicial
236         self._set_vlc_volume(self.volumen.get())
237
238         # Eventos VLC
239         self.events.event_attach(vlc.EventType.MediaPlayerPlaying, self._ev_playing)
240         self.events.event_attach(vlc.EventType.MediaPlayerStopped, self._ev_stopped)
241         self.events.event_attach(vlc.EventType.MediaPlayerEndReached,
↳ self._ev_end_reached)
242
243         self._log("App iniciada.")
244
245         # ----- Video embebido -----
246         def _bind_player_to_widget(self):
247             self.root.update_idletasks()
248             wid = self.video_frame.winfo_id()
249             try:
250                 if sys.platform.startswith("linux"):
251                     self.player.set_xwindow(wid)

```

```

252         elif sys.platform.startswith("win"):
253             self.player.set_hwnd(wid)
254         elif sys.platform == "darwin":
255             self.player.set_nsobject(wid)
256     except Exception as e:
257         self._log(f"No se pudo vincular el área de video: {e}")
258
259     def _show_placeholder(self, text: Optional[str] = None):
260         if text is not None:
261             self.placeholder.config(text=text)
262         if not getattr(self, "placeholder_visible", False):
263             self.placeholder.place(relx=0.5, rely=0.5, anchor="center")
264             self.placeholder_visible = True
265
266     def _hide_placeholder(self):
267         if getattr(self, "placeholder_visible", False):
268             self.placeholder.place_forget()
269             self.placeholder_visible = False
270
271     def _ev_playing(self, _ev): self.root.after(0, self._hide_placeholder)
272     def _ev_stopped(self, _ev): self.root.after(0, self._show_placeholder)
273
274     # === Comportamiento al terminar la reproducción ===
275     def _ev_end_reached(self, _ev):
276         # Ejecutar en hilo de UI
277         def _do():
278             if self.mode == "tx":
279                 if self.tx_loop_enabled:
280                     # DtPlay ya hace loop; sólo reiniciamos el reproductor
281                     ↪ embebido.
282
283                     # Desde la 2ª reproducción en loop, arrancar en 2s para
284                     ↪ compensar el desfase.
285
286                     self.tx_loop_count = getattr(self, "tx_loop_count", 0) + 1
287                     start_ms = 2000 if self.tx_loop_count >= 1 else 0
288                     self._append_to_log_view(

```

```

285         f"[APP] Fin alcanzado (TX, loop ON): reiniciando reproductor
           ↳ en {start_ms} ms.\n"
286     )
287     try:
288         self.player.stop()
289         self._load_media(self.current_media_path,
           ↳ self.current_program_id)
290         self._bind_player_to_widget()
291         self.player.play()
292         self.root.after(120, lambda: self.player.set_time(start_ms))
293         self._start_update_loop()
294     except Exception as e:
295         self._append_to_log_view(f"[APP] Error al reiniciar en loop:
           ↳ {e}\n")
296     else:
297         # Loop OFF: detener TODO (equivalente a presionar botón
           ↳ Detener)
298         self._append_to_log_view("[APP] Fin alcanzado (TX, loop OFF):
           ↳ deteniendo transmisión.\n")
299         self.detener()
300     else:
301         # En preview u otros modos, comportamiento tradicional.
302         self._show_placeholder()
303     self.root.after(0, _do)
304
305
306
307     # ----- Controles -----
308     def abrir_archivo(self):
309         tipos = [("Medios", "*.ts *.mp4 *.mkv *.avi *.mov *.mp3 *.wav *.flac
           ↳ *.m4a"),
310                 ("Todos los archivos", ".*")]
311         path = filedialog.askopenfilename(title="Selecciona un archivo",
           ↳ filetype=tipos)
312         if not path:

```

```

313         return
314
315     self.lbl_archivo.config(text=os.path.basename(path))
316     self._log(f"Abrir medio: {path}")
317
318     self._reset_seek_state()
319     self._set_mode("idle")
320
321     if path.lower().endswith(".ts"):
322         self.current_media_path = path
323         self.placeholder.config(text=os.path.basename(path))
324         self._load_media(path, program_id=None)
325         self.btn_inicio.config(state="normal")
326         self.btn_tx_start.config(state="normal")
327         self._bind_player_to_widget()
328         self._start_update_loop()
329         self._show_placeholder()
330
331         # Detectar programas por ID y seleccionar automáticamente el Canal 1
332         ↪ (primer ID)
333         self.program_list = self._scan_ts_programs(path)
334         # Establecer el programa actual al primero si existe (para que el botón
335         ↪ muestre "Canal 1")
336         if self.program_list:
337             self.current_program_id = int(self.program_list[0]['program'])
338         else:
339             self.current_program_id = None
340         self._update_prog_button()
341         return
342
343     # No-TS
344     self.current_media_path = path
345     try:
346         media = self.instance.media_new(path)
347         self.player.set_media(media)

```

```
346         self.btn_inicio.config(state="normal")
347         self.btn_tx_start.config(state="disabled")
348         self._bind_player_to_widget()
349         self._start_update_loop()
350         self._show_placeholder(os.path.basename(path))
351         self.program_list = []
352         self.current_program_id = None
353         self._update_prog_button()
354     except Exception as e:
355         self._log(f"Error al cargar el medio: {e}")
356         messagebox.showerror("VLC", f"No se pudo cargar el medio:\n{e}")
357         self.current_media_path = None
358         self.lbl_archivo.config(text="Sin archivo cargado")
359         self.btn_inicio.config(state="disabled")
360         self.btn_tx_start.config(state="disabled")
361         self.program_list = []
362         self.current_program_id = None
363         self._update_prog_button()
364         self._set_mode("idle")
365
366
367     def _load_media(self, path: str, program_id: Optional[int]):
368         media = self.instance.media_new(path)
369         if program_id is not None:
370             media.add_option(f":program={int(program_id)}")
371             self._log(f"Forzando programa TS: {program_id}")
372         self.player.set_media(media)
373
374     def reproducir_desde_inicio(self):
375         if not self.current_media_path:
376             messagebox.showinfo("VLC", "Primero abre un archivo.")
377             return
378         try:
379             self._log("Reproducción desde el inicio (previsualización).")
380             self.player.stop()
```

```

381         self._bind_player_to_widget()
382         self.player.play()
383         self.root.after(120, lambda: self.player.set_time(0))
384         self._start_update_loop()
385         self.is_preview = True
386         self._set_mode("preview")
387         self.btn_detener.config(state="normal")
388     except Exception as e:
389         self._log(f"Error al iniciar reproducción: {e}")
390         messagebox.showerror("VLC", f"No se pudo iniciar la reproducción:\n{e}")
391
392     def _set_mode(self, mode: str):
393         self.mode = mode # "idle" / "preview" / "tx"
394         if mode == "idle":
395             self.btn_detener.config(state="disabled")
396             self.btn_inicio.config(state="normal" if self.current_media_path else
397                 ↪ "disabled")
398             ts_ok = bool(self.current_media_path and
399                 ↪ self.current_media_path.lower().endswith(".ts"))
400             self.btn_tx_start.config(state="normal" if ts_ok else "disabled")
401         elif mode == "preview":
402             self.btn_detener.config(state="normal")
403             self.btn_inicio.config(state="normal")
404             ts_ok = bool(self.current_media_path and
405                 ↪ self.current_media_path.lower().endswith(".ts"))
406             self.btn_tx_start.config(state="normal" if ts_ok else "disabled")
407         elif mode == "tx":
408             self.btn_detener.config(state="normal")
409             self.btn_inicio.config(state="disabled")
410             self.btn_tx_start.config(state="disabled")
411
412     def detener(self):
413         try:
414             # Si hay un inicio de TX programado (delay 1s), cancelarlo
415             if getattr(self, "_pending_tx_after_id", None) is not None:

```

```
413         try:
414             self.root.after_cancel(self._pending_tx_after_id)
415         except Exception:
416             pass
417         self._pending_tx_after_id = None
418
419         # Reiniciar contador de loop del reproductor
420         self.tx_loop_count = 0
421
422         if self.mode == "preview":
423             self._log("Detener (modo previsualización).")
424             self.player.stop()
425             self.duration_ms = 0
426             self.lbl_time.config(text="00:00")
427             self.lbl_dur.config(text="00:00")
428             self.progress_var.set(0.0)
429             self._reset_seek_state()
430             self._show_placeholder()
431             self._set_mode("idle")
432
433         elif self.mode == "tx":
434             self._log("Detener (modo transmisión + reproductor).")
435             try:
436                 self.player.stop()
437             except Exception:
438                 pass
439             proc = self.tx_process
440             if proc:
441                 try:
442                     proc.terminate()
443                 try:
444                     proc.wait(timeout=2)
445                 except subprocess.TimeoutExpired:
446                     proc.kill()
447                     proc.wait(timeout=2)
```

```

448         except Exception as e:
449             self._append_to_log_view(f"[DtPlay] error al terminar:
               ↳ {e}\n")
450         finally:
451             self.tx_process = None
452             self.duration_ms = 0
453             self.lbl_time.config(text="00:00")
454             self.lbl_dur.config(text="00:00")
455             self.progress_var.set(0.0)
456             self._reset_seek_state()
457             self._show_placeholder()
458             self._set_mode("idle")
459
460     else:
461         self.btn_detener.config(state="disabled")
462
463     except Exception as e:
464         self._log(f"Error al detener: {e}")
465         messagebox.showerror("VLC", f"No se pudo detener:\n{e}")
466
467     def _reset_seek_state(self):
468         self.is_preview = False
469         self.dragging_seek = False
470
471     # --- Volumen ---
472     def _on_vol_change(self, _val):
473         vol = int(float(_val))
474         self.volumen.set(vol)
475         self.lbl_vol.config(text=f"{vol}%")
476         self._set_vlc_volume(vol)
477
478     def _set_vlc_volume(self, vol):
479         try:
480             self.player.audio_set_volume(int(vol))
481         except Exception:

```

```
482         pass
483
484     # ----- Seek interactivo (solo en previsualización) -----
485     def _seek_value_to_ms(self, val: float) -> int:
486         try:
487             pos = max(0.0, min(1.0, float(val) / 1000.0))
488         except Exception:
489             pos = 0.0
490
491         dur = self.duration_ms if (self.duration_ms and self.duration_ms > 0) else
492         ↪ self.player.get_length()
493         if dur is None or dur <= 0:
494             return 0
495         return int(pos * dur)
496
497     def _on_seek_start(self, _event):
498         if not self.is_preview:
499             return "break"
500
501         if self.duration_ms and self.duration_ms > 0:
502             self.dragging_seek = True
503
504     def _on_seek_drag(self, _event):
505         if not self.is_preview:
506             return "break"
507
508         if self.dragging_seek and self.duration_ms > 0:
509             ms = self._seek_value_to_ms(self.progress_var.get())
510             self.lbl_time.config(text=_fmt_ms(ms))
511
512     def _on_seek_release(self, _event):
513         if not self.is_preview:
514             return "break"
515
516         if not self.dragging_seek:
517             return
518
519         self.dragging_seek = False
520
521         if self.duration_ms and self.duration_ms > 0:
522             target = self._seek_value_to_ms(self.progress_var.get())
```

```
516         try:
517             self.player.set_time(target)
518         except Exception:
519             pass
520
521     # --- Progreso (actualización periódica) ---
522     def _start_update_loop(self):
523         if self.update_job is not None:
524             self.root.after_cancel(self.update_job)
525             self._update_ui()
526
527     def _update_ui(self):
528         length = self.player.get_length()
529         if length and length > 0 and length != self.duration_ms:
530             self.duration_ms = length
531             self.lbl_dur.config(text=_fmt_ms(length))
532
533         cur = self.player.get_time()
534         if cur is None or cur < 0:
535             cur = 0
536         self.lbl_time.config(text=_fmt_ms(cur))
537
538         if self.duration_ms > 0:
539             pos = max(0.0, min(1.0, cur / self.duration_ms))
540             if not self.dragging_seek:
541                 self.progress_var.set(pos * 1000.0)
542         else:
543             pos = self.player.get_position()
544             if pos is not None and pos >= 0 and not self.dragging_seek:
545                 self.progress_var.set(pos * 1000.0)
546             else:
547                 if not self.dragging_seek:
548                     self.progress_var.set(0.0)
549
550         self.update_job = self.root.after(200, self._update_ui)
```

```
551
552     # --- Panel de logs ---
553     def toggle_logs(self):
554         if self.logs_visible:
555             # Ocultar logs
556             self._stop_tail()
557             self.log_frame.pack_forget()
558             self.logs_visible = False
559             self.btn_logs.config(text="Logs ")
560             if self.size_before_logs:
561                 w, h = self.size_before_logs
562                 self.root.geometry(f"{w}x{h}")
563                 self.root.minsize(self.base_min_w, self.base_min_h)
564             self.size_before_logs = None
565         else:
566             self.size_before_logs = (self.root.winfo_width(),
567                                     ↪ self.root.winfo_height())
568             self.log_frame.pack(fill="both", expand=False, pady=(self.LOGS_PADY, 0))
569             self.root.update_idletasks()
570             logs_h = self.log_frame.winfo_reqheight() + self.LOGS_PADY
571             w, h = self.size_before_logs
572             self.root.geometry(f"{w}x{h + logs_h}")
573             self.logs_visible = True
574             self.btn_logs.config(text="Logs ")
575             self._start_tail()
576             self.root.minsize(self.base_min_w, self.base_min_h + logs_h)
577
578     def _start_tail(self):
579         try:
580             with open(self.log_path, "r", encoding="utf-8", errors="replace") as f:
581                 f.seek(0, os.SEEK_END)
582                 self.log_file_pos = f.tell()
583             except FileNotFoundError:
584                 self.log_file_pos = 0
585             self._poll_logs()
```

```
585
586     def _stop_tail(self):
587         if self.log_job is not None:
588             try:
589                 self.root.after_cancel(self.log_job)
590             except Exception:
591                 pass
592             self.log_job = None
593
594     def _poll_logs(self):
595         data = ""
596         try:
597             with open(self.log_path, "r", encoding="utf-8", errors="replace") as f:
598                 f.seek(self.log_file_pos)
599                 data = f.read()
600                 self.log_file_pos = f.tell()
601         except FileNotFoundError:
602             pass
603
604         if data:
605             self._append_to_log_view(data)
606
607         if self.logs_visible:
608             self.log_job = self.root.after(400, self._poll_logs)
609
610     # ===== Cola de logs + drenaje periódico =====
611     def _append_to_log_view(self, text: str):
612         """Productora: encola texto. No toca widgets desde hilos secundarios."""
613         if not isinstance(text, str):
614             text = str(text)
615         chunk = 4000
616         for i in range(0, len(text), chunk):
617             self.log_queue.put(text[i:i + chunk])
618         if self._log_drain_job is None:
619             self._schedule_log_drain(0)
```

```
620
621     def _schedule_log_drain(self, delay_ms: int = 0):
622         self._log_drain_job = self.root.after(delay_ms, self._drain_log_queue)
623
624     def _drain_log_queue(self):
625         self._log_drain_job = None
626         MAX_CHARS_PER_TICK = 20_000
627         emitted = 0
628         if self.log_queue.empty():
629             return
630         self.log_text.configure(state="normal")
631         while not self.log_queue.empty() and emitted < MAX_CHARS_PER_TICK:
632             s = self.log_queue.get_nowait()
633             emitted += len(s)
634             self.log_text.insert("end", s)
635             self._current_log_chars += len(s)
636         if self._current_log_chars > self.MAX_LOG_CHARS:
637             for _ in range(200):
638                 if self._current_log_chars <= self.MAX_LOG_CHARS or
639                    ↪ self.log_text.index("end-1c") == "1.0":
640                     break
641                 first_line_len = int(self.log_text.count("1.0", "2.0", "chars")[0])
642                 self.log_text.delete("1.0", "2.0")
643                 self._current_log_chars -= first_line_len
644             self.log_text.see("end")
645             self.log_text.configure(state="disabled")
646             if not self.log_queue.empty():
647                 self._schedule_log_drain(50)
648
649     def _log(self, msg: str):
650         self._append_to_log_view(f"[APP] {msg}\n")
651
652     def clear_logs(self):
653         try:
654             while not self.log_queue.empty():
```

```

654         self.log_queue.get_nowait()
655     except Exception:
656         pass
657     self.log_text.configure(state="normal")
658     self.log_text.delete("1.0", "end")
659     self.log_text.configure(state="disabled")
660     self._current_log_chars = 0
661
662     # ===== Selector de Programa (MPEG-TS) =====
663     def abrir_selector_programa(self):
664         # Botón ya viene bloqueado si no hay TS; por seguridad, revalidamos
665         if not self.current_media_path or not
        ↪ self.current_media_path.lower().endswith(".ts"):
666             messagebox.showinfo("Programa", "Abre primero un archivo .ts.")
667             return
668
669         # Re-escanear programas (solo por ID)
670         self.program_list = self._scan_ts_programs(self.current_media_path)
671
672         win = tk.Toplevel(self.root)
673         win.title("Seleccionar programa (TS)")
674         win.transient(self.root)
675         win.resizable(False, False)
676
677         cont = ttk.Frame(win, padding=10)
678         cont.pack(fill="both", expand=True)
679
680         ttk.Label(cont, text="Programas detectados:").grid(row=0, column=0,
        ↪ sticky="w")
681         lista = tk.Listbox(cont, height=7, exportselection=False)
682         lista.grid(row=1, column=0, sticky="nsew")
683         cont.grid_rowconfigure(1, weight=1)
684         cont.grid_columnconfigure(0, weight=1)
685
686         # Mostrar solo "Canal N" (ocultar ID al usuario)

```

```
687         for idx, d in enumerate(self.program_list, start=1):
688             lista.insert("end", f"Canal {idx}")
689
690         frm_manual = ttk.Frame(cont)
691         frm_manual.grid(row=2, column=0, sticky="ew", pady=(8, 0))
692         ttk.Label(frm_manual, text="ID manual (program number):").pack(side="left")
693         ent = ttk.Entry(frm_manual, width=10)
694         ent.pack(side="left", padx=(6, 0))
695
696         def rescan():
697             lista.delete(0, "end")
698             self.program_list = self._scan_ts_programs(self.current_media_path)
699             for idx, d in enumerate(self.program_list, start=1):
700                 lista.insert("end", f"Canal {idx}")
701             self._log("[APP] Re-escaneo de programas listo.")
702
703         ttk.Button(frm_manual, text="Re-escanear",
704             ↪ command=rescan).pack(side="right")
705
706         btns = ttk.Frame(cont)
707         btns.grid(row=3, column=0, sticky="e", pady=(10, 0))
708
709         def aplicar():
710             sel = lista.curselection()
711             prog_id: Optional[int] = None
712             if sel:
713                 i = sel[0]
714                 if 0 <= i < len(self.program_list):
715                     prog_id = int(self.program_list[i]['program'])
716             else:
717                 val = ent.get().strip()
718                 if val:
719                     if not val.isdigit():
720                         messagebox.showerror("Programa", "El ID debe ser numérico.")
721                     return
```

```

721         prog_id = int(val)
722         self._aplicar_programa(prog_id)
723         win.destroy()
724
725         # Invertido: Aplicar luego Cancelar
726         btn_aplicar = ttk.Button(btns, text="Aplicar", command=aplicar)
727         btn_aplicar.pack(side="right")
728         btn_cancelar = ttk.Button(btns, text="Cancelar", command=win.destroy)
729         btn_cancelar.pack(side="right", padx=(0, 8))
730
731         # Selección inicial según current_program_id
732         if self.current_program_id is not None:
733             for idx, d in enumerate(self.program_list):
734                 if d['program'] == self.current_program_id:
735                     lista.selection_set(idx)
736                     break
737
738         win.grab_set()
739         win.focus_force()
740
741
742
743     def _scan_ts_programs(self, path: str) -> List[Dict[str, int]]:
744         """
745         Detecta programas de un TS **solo por ID de programa** (program_number),
746         sin intentar inferir PIDs de audio/video.
747         Devuelve una lista de dicts: [{'program': <id>}, ...] ordenados
748         ↪ ascendente.
749         """
750         out: List[Dict[str, int]] = []
751         try:
752             r = run_cmd(["tstables", path])
753             if r.returncode != 0 or not (r.stdout or r.stderr):
754                 return out
755             text = (r.stdout or "") + (r.stderr or "")

```

```

755         # Buscar todos los program_number presentes
756         prog_re = re.compile(r"\bprogram(?:_number)?\s*[:=]\s*(\d+)",
757                               ↪ re.IGNORECASE)
757         ids: Set[int] = set(int(m.group(1)) for m in prog_re.finditer(text))
758         out = [{"program": p} for p in sorted(ids)]
759         return out
760     except FileNotFoundError:
761         self._append_to_log_view("[APP] Aviso: 'tstables' no encontrado. No se
762                               ↪ pueden detectar programas.\n")
763         return out
764     except Exception as e:
765         self._append_to_log_view(f"[APP] Error escaneando TS con tstables:
766                               ↪ {e}\n")
767         return out
768
769     def _aplicar_programa(self, program_id: Optional[int]):
770         """Cambiar de programa sin forzar reproducción si estaba detenida.
771         Si está reproduciendo (o en TX), adelanta 1 segundo para compensar
772         ↪ salto."""
773         if program_id is None:
774             return
775         state = self.player.get_state()
776         was_playing = (state == vlc.State.Playing or self.mode == "tx")
777         was_paused = (state == vlc.State.Paused)
778         cur_ms = self.player.get_time() or 0
779         if cur_ms < 0:
780             cur_ms = 0
781
782         # Si está reproduciendo o transmitiendo, adelantar 1s al cambiar de canal
783         target_ms = cur_ms + 2000 if was_playing else cur_ms
784
785         self.current_program_id = program_id
786         self._update_prog_button()

```

```

786         if not self.current_media_path:
787             return
788
789         try:
790             self.player.stop()
791             self._load_media(self.current_media_path, self.current_program_id)
792             self._bind_player_to_widget()
793
794             if was_playing or was_paused:
795                 self.player.play()
796
797             def seek_and_resume():
798                 try:
799                     # Limitar al largo si ya lo conocemos
800                     dur = self.player.get_length()
801                     t = target_ms
802                     if dur and dur > 0 and t > dur - 1:
803                         t = max(0, dur - 1)
804                     self.player.set_time(t)
805                 except Exception:
806                     pass
807                 if was_paused and not was_playing:
808                     self.player.set_pause(1)
809
810             self.root.after(150, seek_and_resume)
811             self._start_update_loop()
812         else:
813             self._start_update_loop()
814
815     except Exception as e:
816         self._log(f"Error al aplicar programa: {e}")
817         messagebox.showerror("VLC", f"No se pudo aplicar el programa:\n{e}")
818
819
820     def _update_prog_button(self):

```

```

821         """Actualiza texto y estado del botón Programa según contexto (ocultando ID
            ↪ al usuario)."""
822         ts_loaded = bool(self.current_media_path and
            ↪ self.current_media_path.lower().endswith(".ts"))
823         count = len(self.program_list) if ts_loaded else 0
824
825         if not ts_loaded:
826             self.btn_prog.config(text="Programa (0): ---", state="disabled")
827             return
828
829         self.btn_prog.config(state="normal")
830
831         if self.current_program_id is None:
832             # Si hay programas detectados pero no se ha fijado uno, mostrar sólo el
            ↪ conteo
833             self.btn_prog.config(text=f"Programa ({count}): ---")
834         else:
835             # Buscar el índice (Canal N) del ID actual y mostrar sólo "Canal N"
836             idx = None
837             for i, d in enumerate(self.program_list):
838                 if d["program"] == self.current_program_id:
839                     idx = i + 1
840                     break
841             if idx is None:
842                 self.btn_prog.config(text=f"Programa ({count}): ---")
843             else:
844                 self.btn_prog.config(text=f"Programa ({count}): Canal {idx}")
845
846
847
848         # ===== Transmisión (DtPlay) =====
849         def _ask_tx_params(self) -> Optional[dict]:
850             """Dialog para solicitar parámetros de transmisión antes de iniciar
            ↪ DtPlay."""

```

```

851     if not (self.current_media_path and
852             ↪ self.current_media_path.lower().endswith(".ts")):
853         messagebox.showinfo("Iniciar", "Abre primero un archivo .ts para
854             ↪ transmitir.")
855         return None
856
857     dlg = tk.Toplevel(self.root)
858     dlg.title("Parámetros de transmisión (DtPlay)")
859     dlg.transient(self.root)
860     dlg.resizable(False, False)
861
862     frm = ttk.Frame(dlg, padding=10)
863     frm.pack(fill="both", expand=True)
864
865     params = {
866         "t": ("Tipo de tarjeta (-t)", "115"),
867         "i": ("Índice de dispositivo (-i)", "2"),
868         "mf": ("Frecuencia (MHz) (-mf)", "527.143"),
869         "m": ("Tamaño paquete (bytes) (-m)", "188"),
870         "r": ("Bitrate TS (bps) (-r)", "31457280"),
871         "mt": ("Modo transmisión (-mt)", "ISDBT"),
872         "l": ("Nivel log (0=quiet) (-l)", "0"),
873     }
874
875     entries = {}
876     base_freq = 473.143
877     channels_labels: List[str] = []
878     label_to_freq: Dict[str, str] = {}
879     for idx, ch in enumerate(range(14,70)):
880         f = base_freq + 6.0 * idx
881         label = f"Canal {ch} ({f:.3f})"
882         channels_labels.append(label)
883         label_to_freq[label] = f"{f:.3f}"
884
885     row = 0

```

```

884     for key, (label, default) in params.items():
885         ttk.Label(frm, text=f"{label}:").grid(row=row, column=0, sticky="w",
        ↪   pady=3)
886         if key == "mf":
887             cb = ttk.Combobox(frm, state="readonly", width=22,
        ↪   values=channels_labels)
888             try:
889                 default_val = float(default)
890                 match_label = None
891                 for lab in channels_labels:
892                     if abs(float(label_to_freq[lab]) - default_val) < 1e-3:
893                         match_label = lab
894                         break
895                 cb.set(match_label if match_label else channels_labels[0])
896             except Exception:
897                 cb.set(channels_labels[0])
898             cb.grid(row=row, column=1, sticky="w", padx=6)
899             entries[key] = cb
900         else:
901             e = ttk.Entry(frm, width=20)
902             e.insert(0, default)
903             e.grid(row=row, column=1, sticky="w", padx=6)
904             entries[key] = e
905         row += 1
906
907     # LOOP: controla SOLO el reproductor embebido cuando termine
908     var_loop = tk.IntVar(value=0)
909     ttk.Checkbutton(frm, text="Loop al terminar (reinicia solo el reproductor)",
        ↪   variable=var_loop)\
910         .grid(row=row, column=0, columnspan=2, sticky="w", pady=(8,0))
911     row += 1
912
913     info = ttk.Label(frm, text="Usa los valores por defecto si no estás
        ↪   seguro.", foreground="#666")
914     info.grid(row=row, column=0, columnspan=2, sticky="w", pady=(6, 0))

```

```

915
916     btns = ttk.Frame(frm)
917     btns.grid(row=row+1, column=0, columnspan=2, sticky="e", pady=(10, 0))
918
919     result = {"ok": False}
920     def ok():
921         mf_label = entries["mf"].get().strip()
922         mf_str = label_to_freq.get(mf_label, "527.143")
923         try:
924             vals = {
925                 "t": int(entries["t"].get().strip() or "115"),
926                 "i": int(entries["i"].get().strip() or "2"),
927                 "mf": float(mf_str),
928                 "m": int(entries["m"].get().strip() or "188"),
929                 "r": int(entries["r"].get().strip() or "31457280"),
930                 "mt": entries["mt"].get().strip() or "ISDBT",
931                 "l": int(entries["l"].get().strip() or "0"),
932                 "loop": bool(var_loop.get()),
933             }
934         except Exception as e:
935             messagebox.showerror("DtPlay", f"Parámetros inválidos: {e}")
936             return
937         result["ok"] = True
938         result["vals"] = vals
939         dlg.destroy()
940
941     ttk.Button(btns, text="Cancelar", command=dlg.destroy).pack(side="right")
942     ttk.Button(btns, text="Iniciar transmisión", command=ok).pack(side="right",
943 ↪     padx=(0, 8))
944
945     dlg.grab_set()
946     dlg.wait_window()
947     if result.get("ok"):
948         return result["vals"]
949     return None

```

```
949
950     def tx_iniciar(self):
951         vals = self._ask_tx_params()
952         if not vals:
953             return
954
955         if not self.current_media_path or not
↪ self.current_media_path.lower().endswith(".ts"):
956             messagebox.showinfo("Iniciar", "Abre primero un archivo .ts para
↪ transmitir.")
957             return
958         if self.tx_process:
959             messagebox.showinfo("Iniciar", "La transmisión ya está en curso.")
960             return
961
962         # Guardar flag de loop (coordina SOLO el reproductor embebido al finalizar)
963         self.tx_loop_enabled = bool(vals.get("loop", False))
964         self.tx_loop_count = 0 # contador de vueltas del reproductor en modo loop
965
966         ruta_dtplay = self.app_dir
967         cmd = ["/DtPlay", os.path.abspath(self.current_media_path),
968             "-t", str(vals["t"]), "-i", str(vals["i"]),
969             "-mf", str(vals["mf"]), "-m", str(vals["m"]),
970             "-r", str(vals["r"]), "-mt", str(vals["mt"]), "-l", str(vals["l"])]
971
972         # 1) Reproducción embebida primero
973         try:
974             self._log("Reproducción local (embebido) desde el inicio; TX
↪ comenzará en 2s.")
975             self.player.stop()
976             self._load_media(self.current_media_path, self.current_program_id)
977             self._bind_player_to_widget()
978             self.player.play()
979             self.root.after(120, lambda: self.player.set_time(0))
980             self._start_update_loop()
```

```

981         self._reset_seek_state()
982     except Exception as e:
983         self._log(f"Error al reproducir embebido previo a 'Iniciar': {e}")
984
985     # Bloquear 'Iniciar' de inmediato, habilitar 'Detener'
986     self.btn_tx_start.config(state="disabled")
987     self.btn_detener.config(state="normal")
988     # Mantener modo en 'preview' hasta que arranque DtPlay realmente
989
990     # 2) Programar inicio de TX con 2 segundos de retraso (para que el player
991     ↪ vaya por delante)
992     def _launch_tx():
993         # Si el usuario detuvo antes de que dispare, no iniciar
994         if getattr(self, "_pending_tx_after_id", None) is None:
995             return
996         self._pending_tx_after_id = None
997         try:
998             with open(os.devnull, "w") as devnull:
999                 self.tx_process = subprocess.Popen(
1000                     cmd, cwd=ruta_dtplay,
1001                     stdout=devnull, stderr=devnull, text=False, bufsize=0
1002                 )
1003                 self._log("[TX] Transmisión iniciada con DtPlay (después de 2s).")
1004                 self._set_mode("tx")
1005         except Exception as e:
1006             self._log(f"[TX] Error al iniciar DtPlay: {e}")
1007             messagebox.showerror("Iniciar", f"No se pudo iniciar DtPlay:\n{e}")
1008
1009     # Guardar el after-id para poder cancelarlo si se presiona Detener antes
1010     self._pending_tx_after_id = self.root.after(2000, _launch_tx)
1011
1012     # ===== Asistente Crear TS (wizard) =====
1013     def crear_ts_dialog(self, prefill_path1: Optional[str] = None):
1014         win = tk.Toplevel(self.root)
1015         win.title("Crear archivo .ts")

```

```
1015         win.transient(self.root)
1016         win.geometry("720x720")
1017
1018         nb = ttk.Notebook(win)
1019         nb.pack(fill="both", expand=True, padx=8, pady=8)
1020
1021         # ----- Pestaña 1: Fuentes -----
1022         tab_src = ttk.Frame(nb)
1023         nb.add(tab_src, text="Fuentes")
1024
1025         frm = ttk.Frame(tab_src, padding=8)
1026         frm.pack(fill="both", expand=True)
1027
1028         ttk.Label(frm, text="Archivo 1 (mp4/avi):").grid(row=0, column=0,
1029 ↪         sticky="w")
1030         e_src1 = ttk.Entry(frm, width=60)
1031         e_src1.grid(row=0, column=1, sticky="ew", padx=6)
1032         btn_b1 = ttk.Button(frm, text="...", width=3)
1033         btn_b1.grid(row=0, column=2)
1034
1035         var_second = tk.IntVar(value=0)
1036         chk_second = ttk.Checkbutton(frm, text="Usar segundo archivo",
1037 ↪         variable=var_second)
1038         chk_second.grid(row=1, column=0, sticky="w", pady=(8, 0))
1039
1040         ttk.Label(frm, text="Archivo 2 (opcional):").grid(row=2, column=0,
1041 ↪         sticky="w")
1042         e_src2 = ttk.Entry(frm, width=60, state="disabled")
1043         e_src2.grid(row=2, column=1, sticky="ew", padx=6)
1044         btn_b2 = ttk.Button(frm, text="...", width=3)
1045         btn_b2.grid(row=2, column=2)
1046
1047         ttk.Label(frm, text="Carpeta destino:").grid(row=3, column=0, sticky="w",
1048 ↪         pady=(8, 0))
1049         e_dst = ttk.Entry(frm, width=60, state="readonly")
```

```

1046     e_dst.grid(row=3, column=1, sticky="ew", padx=6)
1047     ttk.Button(frm, text="...", width=3, state="disabled").grid(row=3, column=2)
1048
1049     frm.grid_columnconfigure(1, weight=1)
1050
1051     def compute_dst() -> str:
1052         src1 = e_src1.get().strip()
1053         src2 = e_src2.get().strip() if var_second.get() else ""
1054         if not src1:
1055             return ""
1056         d = os.path.dirname(src1)
1057         stem1 = os.path.splitext(os.path.basename(src1))[0]
1058         if var_second.get() and src2:
1059             stem2 = os.path.splitext(os.path.basename(src2))[0]
1060             folder = f"{stem1}+{stem2}"
1061         else:
1062             folder = stem1
1063         return os.path.join(d, folder)
1064
1065     def set_children_state(widget, state: str):
1066         for child in widget.winfo_children():
1067             try:
1068                 child.configure(state=state)
1069             except Exception:
1070                 pass
1071
1072     def browse(entry: ttk.Entry, exts=("*.mp4 *.avi",)):
1073         ft = [("Videos", exts[0]), ("Todos", ".*.*")]
1074         p = filedialog.askopenfilename(title="Seleccionar archivo",
1075             ↪ filetypes=ft)
1076         if p:
1077             entry.delete(0, "end")
1078             entry.insert(0, p)
1079             update_dst()

```

```

1080     # ----- Pestaña 2: A/V + PIDs -----
1081     tab_av = ttk.Frame(nb)
1082     frm2 = ttk.Frame(tab_av, padding=8)
1083
1084     lf1 = ttk.LabelFrame(frm2, text="Programa 1")
1085     ttk.Label(lf1, text="Resolución (-s):").grid(row=0, column=0, sticky="w")
1086     cb_res1 = ttk.Combobox(lf1, values=["720x480", "1280x720", "1920x1080"],
1087     ↪ state="readonly")
1088
1089     cb_res1.set("1920x1080"); cb_res1.grid(row=0, column=1, sticky="w")
1090
1091     ttk.Label(lf1, text="FPS (-r):").grid(row=0, column=2, sticky="w", padx=(12,
1092     ↪ 0))
1093     cb_fps1 = ttk.Combobox(lf1, values=["25", "29.97", "30"], state="readonly")
1094     cb_fps1.set("25"); cb_fps1.grid(row=0, column=3, sticky="w")
1095
1096     ttk.Label(lf1, text="Aspect (-aspect):").grid(row=1, column=0, sticky="w")
1097     cb_aspect1 = ttk.Combobox(lf1, values=["4:3", "16:9"], state="readonly")
1098     cb_aspect1.set("16:9"); cb_aspect1.grid(row=1, column=1, sticky="w")
1099
1100     ttk.Label(lf1, text="Bitrate video (kbps):").grid(row=1, column=2,
1101     ↪ sticky="w", padx=(12, 0))
1102     e_vbit1 = ttk.Entry(lf1, width=8); e_vbit1.insert(0, "8000");
1103     ↪ e_vbit1.grid(row=1, column=3, sticky="w")
1104
1105     ttk.Label(lf1, text="PID video (32-8191):").grid(row=2, column=0,
1106     ↪ sticky="w", pady=(8, 0))
1107     e_pidv1 = ttk.Entry(lf1, width=8); e_pidv1.insert(0, "2064");
1108     ↪ e_pidv1.grid(row=2, column=1, sticky="w", pady=(8, 0))
1109
1110     ttk.Label(lf1, text="PID audio (32-8191):").grid(row=2, column=2,
1111     ↪ sticky="w", padx=(12, 0), pady=(8, 0))
1112     e_pida1 = ttk.Entry(lf1, width=8); e_pida1.insert(0, "2083");
1113     ↪ e_pida1.grid(row=2, column=3, sticky="w", pady=(8, 0))
1114
1115     ttk.Label(lf1, text="Bitrate audio:").grid(row=3, column=0, sticky="w")

```

```

1107     cb_ab1 = ttk.Combobox(lf1, values=[
1108         "32000", "48000", "56000", "64000", "80000", "96000", "112000", "128000",
1109         "160000", "192000", "224000", "256000", "320000", "384000"
1110     ], state="readonly")
1111     cb_ab1.set("128000"); cb_ab1.grid(row=3, column=1, sticky="w")
1112
1113     ttk.Label(lf1, text="Frecuencia (Hz):").grid(row=3, column=2, sticky="w",
1114         ↪ padx=(12, 0))
1115     cb_fs1 = ttk.Combobox(lf1, values=["32000", "44100", "48000"],
1116         ↪ state="readonly")
1117     cb_fs1.set("48000"); cb_fs1.grid(row=3, column=3, sticky="w")
1118
1119     ttk.Label(lf1, text="Offset PTS audio:").grid(row=4, column=0, sticky="w",
1120         ↪ pady=(8, 0))
1121     e_pts1 = ttk.Entry(lf1, width=10); e_pts1.insert(0, "3600");
1122     ↪ e_pts1.grid(row=4, column=1, sticky="w", pady=(8, 0))
1123
1124     ttk.Label(lf1, text="Loop video:").grid(row=4, column=2, sticky="w",
1125         ↪ padx=(12, 0), pady=(8, 0))
1126     var_loop1 = tk.IntVar(value=0)
1127     ttk.Checkbutton(lf1, variable=var_loop1, text="Sí").grid(row=4, column=3,
1128         ↪ sticky="w", pady=(8, 0))
1129
1130     lf2 = ttk.LabelFrame(frm2, text="Programa 2")
1131     ttk.Label(lf2, text="Resolución (-s):").grid(row=0, column=0, sticky="w")
1132     cb_res2 = ttk.Combobox(lf2, values=["720x480", "1280x720", "1920x1080"],
1133         ↪ state="readonly")
1134     cb_res2.set("1920x1080"); cb_res2.grid(row=0, column=1, sticky="w")
1135
1136     ttk.Label(lf2, text="FPS (-r):").grid(row=0, column=2, sticky="w", padx=(12,
1137         ↪ 0))
1138     cb_fps2 = ttk.Combobox(lf2, values=["25", "29.97", "30"], state="readonly")
1139     cb_fps2.set("25"); cb_fps2.grid(row=0, column=3, sticky="w")
1140
1141     ttk.Label(lf2, text="Aspect (-aspect):").grid(row=1, column=0, sticky="w")

```

```

1134     cb_aspect2 = ttk.Combobox(lf2, values=["4:3", "16:9"], state="readonly")
1135     cb_aspect2.set("16:9"); cb_aspect2.grid(row=1, column=1, sticky="w")
1136
1137     ttk.Label(lf2, text="Bitrate video (kbps):").grid(row=1, column=2,
1138     ↪ sticky="w", padx=(12, 0))
1139
1140     e_vbit2 = ttk.Entry(lf2, width=8); e_vbit2.insert(0, "8000");
1141     ↪ e_vbit2.grid(row=1, column=3, sticky="w")
1142
1143     ttk.Label(lf2, text="PID video (32-8191):").grid(row=2, column=0,
1144     ↪ sticky="w", pady=(8, 0))
1145
1146     e_pidv2 = ttk.Entry(lf2, width=8); e_pidv2.insert(0, "2084");
1147     ↪ e_pidv2.grid(row=2, column=1, sticky="w", pady=(8, 0))
1148
1149     ttk.Label(lf2, text="PID audio (32-8191):").grid(row=2, column=2,
1150     ↪ sticky="w", padx=(12, 0), pady=(8, 0))
1151
1152     e_pida2 = ttk.Entry(lf2, width=8); e_pida2.insert(0, "2103");
1153     ↪ e_pida2.grid(row=2, column=3, sticky="w", pady=(8, 0))
1154
1155     ttk.Label(lf2, text="Bitrate audio:").grid(row=3, column=0, sticky="w")
1156
1157     cb_ab2 = ttk.Combobox(lf2, values=[
1158         "32000", "48000", "56000", "64000", "80000", "96000", "112000", "128000",
1159         "160000", "192000", "224000", "256000", "320000", "384000"
1160     ], state="readonly")
1161     cb_ab2.set("128000"); cb_ab2.grid(row=3, column=1, sticky="w")
1162
1163     ttk.Label(lf2, text="Frecuencia (Hz):").grid(row=3, column=2, sticky="w",
1164     ↪ padx=(12, 0))
1165
1166     cb_fs2 = ttk.Combobox(lf2, values=["32000", "44100", "48000"],
1167     ↪ state="readonly")
1168     cb_fs2.set("48000"); cb_fs2.grid(row=3, column=3, sticky="w")
1169
1170     ttk.Label(lf2, text="Offset PTS audio:").grid(row=4, column=0, sticky="w",
1171     ↪ pady=(8, 0))
1172
1173     e_pts2 = ttk.Entry(lf2, width=10); e_pts2.insert(0, "3600");
1174     ↪ e_pts2.grid(row=4, column=1, sticky="w", pady=(8, 0))

```

```

1159
1160     ttk.Label(lf2, text="Loop video:").grid(row=4, column=2, sticky="w",
↪     padx=(12, 0), pady=(8, 0))
1161     var_loop2 = tk.IntVar(value=0)
1162     ttk.Checkbutton(lf2, variable=var_loop2, text="Sí").grid(row=4, column=3,
↪     sticky="w", pady=(8, 0))
1163
1164     lf1.grid(row=0, column=0, sticky="nsew", pady=(0, 8))
1165     lf2.grid(row=1, column=0, sticky="nsew")
1166     frm2.grid_columnconfigure(0, weight=1)
1167
1168     # ----- Pestaña 3: Nombres/Tablas -----
1169     tab_names = ttk.Frame(nb)
1170     frm3 = ttk.Frame(tab_names, padding=8)
1171
1172     ttk.Label(frm3, text="Nombre de la Red:").grid(row=0, column=0, sticky="w")
1173     e_net = ttk.Entry(frm3, width=30); e_net.insert(0, "OdiseaStreamer");
↪     e_net.grid(row=0, column=1, sticky="w")
1174     ttk.Label(frm3, text="Canal 1:").grid(row=1, column=0, sticky="w", pady=(6,
↪     0))
1175     e_ch1 = ttk.Entry(frm3, width=30); e_ch1.insert(0, "Canal Odisea");
↪     e_ch1.grid(row=1, column=1, sticky="w", pady=(6, 0))
1176     ttk.Label(frm3, text="Programa 1:").grid(row=2, column=0, sticky="w")
1177     e_pr1 = ttk.Entry(frm3, width=30); e_pr1.insert(0, "Programa Odisea");
↪     e_pr1.grid(row=2, column=1, sticky="w")
1178     ttk.Label(frm3, text="Descripción 1:").grid(row=3, column=0, sticky="w")
1179     e_desc1 = ttk.Entry(frm3, width=40); e_desc1.insert(0, "Descripcion del
↪     programa"); e_desc1.grid(row=3, column=1, sticky="w")
1180     ttk.Label(frm3, text="Canal 2 (si hay segundo archivo:).grid(row=4,
↪     column=0, sticky="w", pady=(6, 0))
1181     e_ch2 = ttk.Entry(frm3, width=30); e_ch2.insert(0, "Canal 2");
↪     e_ch2.grid(row=4, column=1, sticky="w", pady=(6, 0))
1182     ttk.Label(frm3, text="Programa 2:").grid(row=5, column=0, sticky="w")
1183     e_pr2 = ttk.Entry(frm3, width=30); e_pr2.insert(0, "Programa 2");
↪     e_pr2.grid(row=5, column=1, sticky="w")

```

```
1184     ttk.Label(frm3, text="Descripción 2:").grid(row=6, column=0, sticky="w")
1185     e_desc2 = ttk.Entry(frm3, width=40); e_desc2.insert(0, "Descripcion 2");
1186     ↪ e_desc2.grid(row=6, column=1, sticky="w")
1187
1188     # --- Marquee pequeñito dentro de la ÚLTIMA pestaña (feedback de vida) ---
1189     # Será un Canvas con un rectángulo que se mueve de DERECHA a IZQUIERDA
1190     ↪ mientras se genera el TS.
1191     marquee_frame = ttk.Frame(frm3)
1192     marquee_frame.grid(row=7, column=0, columnspan=2, sticky="ew", pady=(16, 0))
1193     frm3.grid_columnconfigure(0, weight=0)
1194     frm3.grid_columnconfigure(1, weight=1)
1195
1196     marquee_canvas = tk.Canvas(marquee_frame, height=6, highlightthickness=0,
1197     ↪ bg=self.root.cget("bg"))
1198     marquee_canvas.pack(fill="x", expand=True)
1199     # Rectángulo móvil
1200     marquee_rect = marquee_canvas.create_rectangle(0, 0, 60, 6, fill="#4a90e2",
1201     ↪ width=0)
1202
1203     marquee_running = {"on": False}
1204     def marquee_reset():
1205         w = marquee_canvas.winfo_width()
1206         marquee_canvas.coords(marquee_rect, w, 0, w + 60, 6)
1207
1208     def marquee_tick():
1209         if not marquee_running["on"]:
1210             return
1211         try:
1212             # mover 8 px por tick hacia la IZQUIERDA
1213             marquee_canvas.move(marquee_rect, -8, 0)
1214             x1, _, x2, _ = marquee_canvas.coords(marquee_rect)
1215             w = marquee_canvas.winfo_width()
1216             # si salió completamente por la izquierda, reaparece a la derecha
1217             if x2 <= 0:
1218                 marquee_canvas.coords(marquee_rect, w, 0, w + 60, 6)
```

```

1215         except Exception:
1216             pass
1217         # 30 ms por tick (ajustable)
1218         marquee_canvas.after(30, marquee_tick)
1219
1220     def marquee_start():
1221         marquee_running["on"] = True
1222         marquee_reset()
1223         marquee_tick()
1224
1225     def marquee_stop():
1226         marquee_running["on"] = False
1227         # dejar el rectángulo quieto (lo movemos al principio y no lanzamos más
1228         ↪ ticks)
1229         try:
1230             marquee_canvas.coords(marquee_rect, 0, 0, 60, 6)
1231         except Exception:
1232             pass
1233
1234     nb.add(tab_av, text="A/V + PIDs"); frm2.pack(fill="both", expand=True)
1235     nb.add(tab_names, text="Nombres/Tablas"); frm3.pack(fill="both",
1236     ↪ expand=True)
1237
1238     # --- Footer (global del diálogo) ---
1239     footer = ttk.Frame(win); footer.pack(fill="x", padx=8, pady=(0, 8))
1240     progbar = ttk.Progressbar(footer, mode="determinate", maximum=100, value=0)
1241     progbar.pack(side="left", fill="x", expand=True, padx=(0, 8))
1242     lbl_pct = ttk.Label(footer, text="0%"); lbl_pct.pack(side="left", padx=(0,
1243     ↪ 12))
1244     btn_close = ttk.Button(footer, text="Cerrar", width=10,
1245     ↪ command=win.destroy); btn_close.pack(side="left")
1246     btn_next = ttk.Button(footer, text="Siguiete"); btn_next.pack(side="right")
1247     btn_go = ttk.Button(footer, text="Generar TS", width=15)
1248
1249     def toggle_block2(enable: bool):

```

```

1246         state = ("normal" if enable else "disabled")
1247         set_children_state(lf2, state)
1248
1249     def toggle_names2(enable: bool):
1250         state = ("normal" if enable else "disabled")
1251         for w in (e_ch2, e_pr2, e_desc2):
1252             try:
1253                 w.configure(state=state)
1254             except Exception:
1255                 pass
1256
1257     def update_dst(*_):
1258         e_src2.configure(state=("normal" if var_second.get() else "disabled"))
1259         dst = compute_dst()
1260         e_dst.configure(state="normal"); e_dst.delete(0, "end"); e_dst.insert(0,
1261             ↪ dst); e_dst.configure(state="readonly")
1262         second_on = (var_second.get() == 1)
1263         toggle_block2(second_on); toggle_names2(second_on)
1264
1265     btn_b1.config(command=lambda: browse(e_src1))
1266     btn_b2.config(command=lambda: browse(e_src2))
1267     var_second.trace_add("write", update_dst)
1268     e_src1.bind("<FocusOut>", update_dst); e_src2.bind("<FocusOut>", update_dst)
1269     if prefill_path1: e_src1.insert(0, prefill_path1)
1270     win.after(0, update_dst)
1271
1272     def on_tab_changed(_evt=None):
1273         idx = nb.index(nb.select())
1274         if idx in (0, 1):
1275             if not btn_next.winfo_ismapped(): btn_next.pack(side="right")
1276             if btn_go.winfo_ismapped(): btn_go.pack_forget()
1277         else:
1278             if btn_next.winfo_ismapped(): btn_next.pack_forget()
1279             if not btn_go.winfo_ismapped(): btn_go.pack(side="right")

```

```

1279     nb.bind("<<NotebookTabChanged>>", on_tab_changed); win.after(0,
        ↪ on_tab_changed)
1280
1281     def validate_sources() -> bool:
1282         src1 = e_src1.get().strip()
1283         use2 = bool(var_second.get())
1284         src2 = e_src2.get().strip() if use2 else ""
1285         if not src1:
1286             messagebox.showerror("Crear TS", "Selecciona el Archivo 1."); return
        ↪ False
1287         if use2 and not src2:
1288             messagebox.showerror("Crear TS", "Selecciona el Archivo 2 o desmarca
        ↪ 'Usar segundo archivo'."); return False
1289         return True
1290
1291     def _read_int(widget: ttk.Entry, name: str) -> Optional[int]:
1292         try: return int(widget.get().strip())
1293         except Exception:
1294             messagebox.showerror("Crear TS", f"{name} debe ser un número.");
        ↪ return None
1295
1296     def _check_pid_range(pid: int, name: str) -> bool:
1297         if not (32 <= pid <= 8191):
1298             messagebox.showerror("Crear TS", f"{name} debe estar en el rango
        ↪ 32..8191."); return False
1299         return True
1300
1301     def validate_av_pids() -> bool:
1302         pidv1 = _read_int(e_pidv1, "PID video (Programa 1)")
1303         pida1 = _read_int(e_pida1, "PID audio (Programa 1)")
1304         if pidv1 is None or pida1 is None: return False
1305         if not _check_pid_range(pidv1, "PID video (Programa 1)": return False
1306         if not _check_pid_range(pida1, "PID audio (Programa 1)": return False
1307         pids = [pidv1, pida1]; labels = ["PID video P1", "PID audio P1"]
1308         if var_second.get():

```

```

1309         pidv2 = _read_int(e_pidv2, "PID video (Programa 2)")
1310         pida2 = _read_int(e_pida2, "PID audio (Programa 2)")
1311         if pidv2 is None or pida2 is None: return False
1312         if not _check_pid_range(pidv2, "PID video (Programa 2)": return
        ↪ False
1313         if not _check_pid_range(pida2, "PID audio (Programa 2)": return
        ↪ False
1314         pids += [pidv2, pida2]; labels += ["PID video P2", "PID audio P2"]
1315     if len(set(pids)) != len(pids):
1316         seen = {}; dups = []
1317         for val, label in zip(pids, labels):
1318             if val in seen: dups.append(f"{label} = {val} dup con
        ↪ {seen[val]}")
1319             else: seen[val] = label
1320         messagebox.showerror("Crear TS", "PIDs repetidos:\n" +
        ↪ "\n".join(dups) + "\n\nTodos los PIDs deben ser distintos.")
1321         return False
1322     return True
1323
1324     def next_tab():
1325         idx = nb.index(nb.select())
1326         if idx == 0:
1327             if not validate_sources(): return
1328             nb.select(1)
1329         elif idx == 1:
1330             if not validate_av_pids(): return
1331             nb.select(2)
1332
1333     btn_next.config(command=next_tab)
1334
1335     # ----- Pipeline -----
1336     def pipeline():
1337         if not validate_sources(): return
1338         if not validate_av_pids(): return
1339

```

```

1340     # Iniciar MARQUEE (feedback) en la última pestaña
1341     self.root.after(0, marquee_start)
1342
1343     try:
1344         src1 = e_src1.get().strip()
1345         use2 = bool(var_second.get())
1346         src2 = e_src2.get().strip() if use2 else ""
1347
1348         dst = os.path.normpath(os.path.join(
1349             os.path.dirname(src1),
1350             (os.path.splitext(os.path.basename(src1))[0]
1351              if not (use2 and src2)
1352              else
1353                 ↪ f"{os.path.splitext(os.path.basename(src1))[0]}+{os.path.splitext(os.path.basename(src1))[1]}"))
1354         os.makedirs(dst, exist_ok=True)
1355
1356         res1 = cb_res1.get(); fps1 = cb_fps1.get(); aspect1 =
1357             ↪ cb_aspect1.get()
1358         try: vbit1 = int(e_vbit1.get().strip())
1359         except Exception: vbit1 = 8000
1360         pidv1 = int(e_pidv1.get().strip()); pida1 =
1361             ↪ int(e_pida1.get().strip())
1362         ab1 = int(cb_ab1.get()); fs1 = int(cb_fs1.get()); pts1 =
1363             ↪ e_pts1.get().strip()
1364         loop1 = 1 if var_loop1.get() else 0
1365
1366         if use2:
1367             res2 = cb_res2.get(); fps2 = cb_fps2.get(); aspect2 =
1368                 ↪ cb_aspect2.get()
1369             try: vbit2 = int(e_vbit2.get().strip())
1370             except Exception: vbit2 = 8000
1371             pidv2 = int(e_pidv2.get().strip()); pida2 =
1372                 ↪ int(e_pida2.get().strip())

```

```

1368         ab2 = int(cb_ab2.get()); fs2 = int(cb_fs2.get()); pts2 =
           ↪ e_pts2.get().strip()
1369         loop2 = 1 if var_loop2.get() else 0
1370
1371     def lg(s): self._append_to_log_view(f"[TS] {s}\n")
1372
1373     steps_programa = 6
1374     total_steps = steps_programa * (2 if use2 else 1)
1375     total_steps += 4 # null.ts + tablas + mux + tsstamp
1376     steps_done = 0
1377
1378     def _set_progress(done: int, total: int, msg: Optional[str] = None):
1379         pct = max(0, min(100, int((done / total) * 100)))
1380         def _ui():
1381             progbar.config(value=pct)
1382             lbl_pct.config(text=f"{pct}%")
1383             if msg:
1384                 self._append_to_log_view(f"[TS] {msg}\n")
1385                 # Si llegó a 100%, detener el MARQUEE y dejarlo quieto
1386                 if pct >= 100:
1387                     marquee_stop()
1388             self.root.after(0, _ui)
1389
1390     def _step_ok(msg: str):
1391         nonlocal steps_done
1392         steps_done += 1
1393         _set_progress(steps_done, total_steps, msg)
1394
1395     def gen_for(src, tag, pid_v, pid_a, res, fps, aspect, vbit, ab, fs,
           ↪ pts, loop):
1396         vbit_s = f"{vbit}k"
1397         audio_mp2 = os.path.join(dst, f"audio{tag}.mp2")
1398         video_m2v = os.path.join(dst, f"video{tag}.m2v")
1399         audio_pes = os.path.join(dst, f"audio{tag}.pes")
1400         video_pes = os.path.join(dst, f"video{tag}.pes")

```

```

1401         audio_ts = os.path.join(dst, f"audio{tag}.ts")
1402         video_ts = os.path.join(dst, f"video{tag}.ts")
1403
1404         cmd = (f'ffmpeg -hide_banner -loglevel error -y -i "{src}" '
1405               f'-vn -ac 2 -acodec mp2 -f mp2 -ab {ab} -ar {fs} '
1406               ↪ f'"{audio_mp2}"')
1407
1408         lg(f"Audio mp2 ({tag})..."); run_cmd_shell_quiet(cmd);
1409         ↪ _step_ok(f"Audio {tag}: ES (mp2) listo")
1410
1411         cmd = (f'ffmpeg -hide_banner -loglevel error -y -i "{src}" -an '
1412               f'-vcodec mpeg2video -f mpeg2video '
1413               f'-s {res} -r {fps} -aspect {aspect} -deinterlace '
1414               f'-b {vbit_s} -minrate {vbit_s} -maxrate {vbit_s} '
1415               f'-bf 2 -bufsize 1835008 -pix_fmt yuv420p "{video_m2v}"')
1416         lg(f"Video m2v ({tag})..."); run_cmd_shell_quiet(cmd);
1417         ↪ _step_ok(f"Video {tag}: ES (m2v) listo")
1418
1419         cmd = f'esvideo2pes "{video_m2v}" > "{video_pes}"'
1420         lg(f"Video PES ({tag})..."); run_cmd_shell_quiet(cmd, cwd=dst);
1421         ↪ _step_ok(f"Video {tag}: PES listo")
1422
1423         frame_size = int(144 * ab / fs)
1424         cmd = f'esaudio2pes "{audio_mp2}" 1152 {fs} {frame_size} {pts} > '
1425         ↪ f'"{audio_pes}"'
1426         lg(f"Audio PES ({tag})..."); run_cmd_shell_quiet(cmd, cwd=dst);
1427         ↪ _step_ok(f"Audio {tag}: PES listo")
1428
1429         vbv = 112; br_ts = int(vbit * 1.15 * 1000)
1430         cmd = f'pesvideo2ts {pid_v} {fps} {vbv} {br_ts} {loop} '
1431         ↪ f'"{video_pes}" > "{video_ts}"'
1432         lg(f"Video TS ({tag})..."); run_cmd_shell_quiet(cmd, cwd=dst);
1433         ↪ _step_ok(f"Video {tag}: TS listo")
1434
1435         cmd = f'pesaudio2ts {pid_a} 1152 {fs} {frame_size} {loop} '
1436         ↪ f'"{audio_pes}" > "{audio_ts}"'

```

```

1427         lg(f"Audio TS ({tag})..."); run_cmd_shell_quiet(cmd, cwd=dst);
        ↪ _step_ok(f"Audio {tag}: TS listo")
1428
1429     gen_for(src1, "1", pidv1, pida1, res1, fps1, aspect1, vbit1, ab1,
        ↪ fs1, pts1, loop1)
1430     if use2:
1431         gen_for(src2, "2", pidv2, pida2, res2, fps2, aspect2, vbit2,
        ↪ ab2, fs2, pts2, loop2)
1432
1433     os_res = os.path.join(self.app_dir, "OS_Resource")
1434     try:
1435         src_null = os.path.join(os_res, "null.ts"); dst_null =
        ↪ os.path.join(dst, "null.ts")
1436         if os.path.isfile(src_null):
1437             with open(src_null, "rb") as fsrc, open(dst_null, "wb") as
        ↪ fdst: fdst.write(fsrc.read())
1438             _step_ok("null.ts copiado")
1439         else:
1440             lg("Aviso: OS_Resource/null.ts no encontrado.")
1441     except Exception as e:
1442         lg(f"Error copiando null.ts: {e}")
1443
1444     if use2:
1445         script = os.path.join(os_res, "generar_tablas_dual.py")
1446         cmd = ["python", script, e_net.get().strip(),
1447             ↪ e_ch1.get().strip(), e_pr1.get().strip(),
1448             ↪ e_desc1.get().strip(),
1449             ↪ e_ch2.get().strip(), e_pr2.get().strip(),
1450             ↪ e_desc2.get().strip(),
1451             ↪ str(pidv1), str(pida1), str(pidv2), str(pida2), dst]
1452     else:
1453         script = os.path.join(os_res, "generar_tablas_py2.py")
1454         cmd = ["python", script, e_net.get().strip(),
1455             ↪ e_ch1.get().strip(),

```

```

1453         str(pidal), str(pidv1), dst, e_pr1.get().strip(),
        ↪ e_desc1.get().strip()]
1454 lg("Generando tablas...")
1455 with open(os.devnull, "w") as devnull:
1456     subprocess.run(cmd, cwd=os_res, stdout=devnull, stderr=devnull,
        ↪ text=False)
1457 _step_ok("Tablas SI/PSI generadas")
1458
1459 pat_bitrate = 15040; pmt_bitrate = 15040; sdt_bitrate = 3008
1460 nit_bitrate = 3008; eit_bitrate = 3008; null_bitrate = 29958294
1461 if use2:
1462     mux = (f"tsmuxer b:{pat_bitrate} pat.ts b:{pmt_bitrate}
        ↪ pmt_sd_1.ts "
1463           f"b:{pmt_bitrate} pmt_sd_2.ts b:{sdt_bitrate} sdt.ts
        ↪ b:{nit_bitrate} nit.ts "
1464           f"b:{eit_bitrate} eit1.ts b:{eit_bitrate} eit2.ts "
1465           f'c:{int(vbit1*1000*1.15)} video1.ts c:188000 audio1.ts '
1466           f'c:{int(vbit2*1000*1.15)} video2.ts c:188000 audio2.ts '
1467           f'o:{null_bitrate} null.ts > semifinal.ts")
1468 else:
1469     mux = (f"tsmuxer b:{pat_bitrate} pat.ts b:{pmt_bitrate}
        ↪ pmt_sd.ts "
1470           f"b:{sdt_bitrate} sdt.ts b:{nit_bitrate} nit.ts
        ↪ b:{eit_bitrate} eit.ts "
1471           f'c:{int(vbit1*1000*1.15)} video1.ts c:188000 audio1.ts '
1472           f'o:{null_bitrate} null.ts > semifinal.ts")
1473 lg("Multiplexando (CBR)..."); run_cmd_shell_quiet(mux, cwd=dst);
        ↪ _step_ok("Multiplex CBR completado")
1474
1475 out_name = os.path.basename(dst) + ".ts"; salida_final =
        ↪ os.path.join(os.path.dirname(dst), out_name)
1476 cmd = f'tsstamp semifinal.ts {null_bitrate} > "{salida_final}"'
1477 lg("Aplicando tsstamp..."); run_cmd_shell_quiet(cmd, cwd=dst);
        ↪ _step_ok("tsstamp aplicado")
1478

```

```

1479         try:
1480             r = run_cmd(["tstables", salida_final])
1481             if r.returncode == 0 and r.stdout:
1482                 self._append_to_log_view("\n[TS] === Resumen tstables
↳ ===\n"); self._append_to_log_view(r.stdout + "\n")
1483         except Exception as e:
1484             self._append_to_log_view(f"[TS] Aviso: tstables no disponible
↳ ({e})\n")
1485
1486         self.current_media_path = salida_final
1487         self._load_media(salida_final, program_id=None)
1488         self._show_placeholder(os.path.basename(salida_final))
1489         self.lbl_archivo.config(text=os.path.basename(salida_final))
1490         self.btn_inicio.config(state="normal");
↳ self.btn_tx_start.config(state="normal")
1491         self._start_update_loop(); self._reset_seek_state()
1492         self._append_to_log_view(f"[TS] Listo: {salida_final}\n")
1493
1494         self.program_list = self._scan_ts_programs(salida_final)
1495         self.current_program_id = None; self._update_prog_button()
1496
1497         _set_progress(total_steps, total_steps, "Conversión finalizada")
1498         self.root.after(0, lambda: (progbar.config(value=100),
↳ lbl_pct.config(text="100%")))
1499         # Parar el marquee definitivamente
1500         self.root.after(0, marquee_stop)
1501         win.after(1000, win.destroy)
1502
1503     finally:
1504         # En caso de error o salida anticipada, asegurar que el marquee no
↳ quede corriendo
1505         self.root.after(0, marquee_stop)
1506
1507     def start_pipeline():
1508         idx = nb.index(nb.select())

```

```

1509         if idx != 2:
1510             return
1511         if not validate_sources():
1512             return
1513         if not validate_av_pids():
1514             return
1515         progbar.config(mode="determinate", value=0, maximum=100);
1516         ↪ lbl_pct.config(text="0%")
1517         btn_go.config(state="disabled")
1518         # Lanzar la animación ya desde el UI (última pestaña visible)
1519         marquee_start()
1520         threading.Thread(target=lambda: (
1521             pipeline(),
1522             self.root.after(0, lambda: btn_go.config(state="normal"))
1523         ), daemon=True).start()
1524
1525         btn_go.config(command=start_pipeline)
1526
1527
1528     def _dump_result(self, r: subprocess.CompletedProcess):
1529         if r.stdout: self._append_to_log_view(r.stdout)
1530         if r.stderr: self._append_to_log_view(r.stderr)
1531         self._append_to_log_view(f"[CMD] rc={r.returncode}\n")
1532
1533     # --- Cierre ---
1534     def on_close(self):
1535         try:
1536             if self.update_job is not None:
1537                 self.root.after_cancel(self.update_job)
1538             self._stop_tail()
1539             self.player.stop()
1540         except Exception:
1541             pass
1542         proc = self.tx_process

```

```
1543         if proc:
1544             try:
1545                 proc.terminate()
1546             try:
1547                 proc.wait(timeout=2)
1548             except subprocess.TimeoutExpired:
1549                 proc.kill()
1550                 proc.wait(timeout=2)
1551         except Exception:
1552             pass
1553         finally:
1554             self.tx_process = None
1555         self.root.destroy()
1556
1557
1558     def main():
1559         root = tk.Tk()
1560         app = VLCApp(root)
1561         root.protocol("WM_DELETE_WINDOW", app.on_close)
1562         root.mainloop()
1563
1564
1565     if __name__ == "__main__":
1566         main()
1567
1568
```