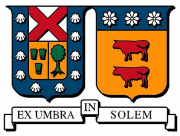

Universidad Técnica Federico Santa María
Campus Casa Central
Valparaíso, Chile

"MANUAL TRANSMISIÓN DE VIDEO UTILIZANDO TARJETA MODULADORA DTA-115 EN LINUX"

Televisión digital bajo norma ISDB-Tb

-

Desarrollado por:
Sebastián Castillo Araya
Correo :
sebastian.castilloar@sansano.usm.cl

Fecha:
28 Diciembre 2022

Índice

Índice de tablas

Índice de figuras

1. Introducción

El presente manual tiene como objetivo guíar al lector paso a paso para transmitir video a la televisión digital, bajo la norma ISDB-Tb, utilizando la tarjeta moduladora DTA-115 desarrollada por DekTec. Se hace énfasis en que esta es una guía que se enfoca plenamente en realizar la transmisión y que no busca explicar conceptos teóricos que pueda conllevar esta. Para casos en que se desee profundizar conceptos se recomienda al lector revisar documentos en sección Referencias los cuales proveen mayor información.

2. Requerimientos

Los requisitos de hardware y software indicados a continuación, fueron utilizados en pruebas de transmisión y lograron un resultado satisfactorio.

2.1. Hardware

Para **transmitir**:

- Computador:
  - CPU: Intel Core i5-2400 3.1 GHz
  - Memoria Ram: 4GB
  - Disco Duro: 250GB
- Tarjeta Moduladora DTA-115.
- Una antena para transmitir.

Para **recibir**(opcional para observar resultados):

- Televisión.
- Set Top-Box (si la televisión ya cuenta con compatibilidad digital entonces este dispositivo no hace falta).
- Una antena para recibir.

2.2. Software

- Sistema operativo GNU/Linux Ubuntu 14.04 LTS
- Python 2.7.6
- GCC 4.8.4
- ffmpeg version N-79177
- DtPlay en Linux
- OpenCaster 2.4-3.2.2 con parche que agregó LIFIA para compatibilidad ISDB-Tb

La mayoría del software utilizado no debería significar problema su adquisición, aún así respecto a los dos últimos requisitos de software se indica como llevar a caso su instalación ya que necesitan modificaciones.

2.2.1. Instalación de DtPLay en Linux

DtPlay es un software para Linux, perteneciente a la empresa Dektec, el cual permite al usuario realizar una transmisión de un transport stream generado por OpenCaster.

Lo primero a realizar es descargar los controladores de la tarjeta moduladora DTA-115 y el SDK para Linux, los cuales se encuentran disponibles en la página oficial:

Drivers

Para instalar los drivers se siguen los siguientes pasos:

Descomprimir las fuente

```
$ tar zxf LinuxSDK.tar.gz
```

Dentro de la carpeta Linux ejecutar comando make

```
$ cd LinuxSDK/Drivers/Dta/Source/Linux
$ make
$ sudo make install
```

Siguiendo con la instalación, se descarga el software DtPlay de la página oficial:

DtPlay para Linux

Se deben copiar los archivos DTAPI.o y DTAPI64.o de la carpeta

`/LinuxSDK/DTAPI/lib/GCC4.3.2`

también el archivo DTAPI.h de la carpeta

`/LinuxSDK/DTAPI/Include`

y los tres archivos pegarlos en la carpeta

`/DtPlay/DTAPI/`

Luego el siguiente paso es ejecutar make en la carpeta DtPlay

```
$ cd DtPlay
$ make
```

Para ejecutar el programa se puede utilizar el siguiente comando:

```
$ ./DtPlay <file> -t 115
```

DtPlay es un software que permite transmitir archivos transport stream que contengan la cabecera Transmission Multiplexing Configuration Control (TMCC), pero los archivos TS generados por OpenCaster no incluyen estos bytes, por lo que es necesario modificar el código fuente del programa.

Para esto se modifica el archivo

`DtPlay.cpp ubicado en DtPlay/Source`

En el que se modifica la función

`InitIsdbtPars`

la cual inicializa los parámetros de la norma ISDB-Tb, por ejemplo el número de segmentos, layers, coderate, modulación, etc.

Los parámetros escogidos son en base a los utilizados por LIFIA en sus test.

Primero se deben comentar las siguientes líneas:

```
// IsdbtPars.m_DoMux = false;
// dr = IsdbtPars.RetrieveParsFromTs(pTempBuf, NumBytes);
```

Y lo siguiente agregar los parametros del archivo transport stream que se desea trasmitir

```
// Cambios para agregar parametros ISDB-T
        IsdbtPars.m_DoMux=true;
        IsdbtPars.m_FilledOut=true;
        IsdbtPars.m_ParXtra0=90413;
        IsdbtPars.m_BType=0;
        IsdbtPars.m_Mode=3;
        IsdbtPars.m_Guard=2;
        IsdbtPars.m_PartialRx=0;
        IsdbtPars.m_Emergency=0;
        IsdbtPars.m_IipPid=8176;
        IsdbtPars.m_LayerOther=1;
        IsdbtPars.m_Virtual13Segm=0;
        DtIsdbtLayerPars  layer1,layer2, layer3;
        layer1.m_NumSegments=13;
        layer1.m_Modulation=3;
        layer1.m_CodeRate=2;
        layer1.m_TimeInterleave=2;
        IsdbtPars.m_LayerPars[0]=layer1;
        layer2.m_NumSegments=0;
        layer2.m_Modulation=3;
        layer2.m_CodeRate=2;
        layer2.m_TimeInterleave=2;
        IsdbtPars.m_LayerPars[1]=layer2;
        layer3.m_NumSegments=0;
        layer3.m_Modulation=3;
        layer3.m_CodeRate=2;
        layer3.m_TimeInterleave=2;
        IsdbtPars.m_LayerPars[2]=layer3;
        IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(0,1)); //PAT
        IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(16,1); //NIT
        IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(17,1); //SDT
           /BAT
        IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(1031,1)); //
           PMT 1031
```

```
33          IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(2001,1));
34          IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(2004,1));
35          IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(2020,1));
36          IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(2063,1)); //
               VIDEO
37          IsdbtPars.m_Pid2Layer.insert(std::pair<int,int>(2083,1)); //
               AUDIO
```

Luego de realizar los cambios se debe compilar el código fuente nuevamente.

```
$ cd DtPlay
$ make
```

2.2.2. Instalación de OpenCaster 2.4-3.2.2 con parche de LIFIA para compatibilidad ISDB-Tb

OpenCaster es un software gratuito y abierto desarrollado por AVALPA, para el cual el Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA), de la Universidad Nacional de la Plata, programó un parche para la versión 2.4 para que este soporte la norma ISDB-Tb. OpenCaster permite generar archivos transport stream que posteriormente pueden ser transmitidos con DtPlay.

OpenCaster puede ser descargado directamente de la página oficial de AVALPA:

Sitio oficial AVALPA

Pero la versión que se encuentra disponible es más actual que la 2.4. Para solucionar esto se adjunta un vínculo en que se puede descargar OpenCaster 2.4 y además el parche programado por LIFIA.

OpenCaster2.4

PatchLIFIA2.4

Para instalar el software se deben ubicar los archivos OpenCaster2.4.tgz y OpenCaster-lifia-rev362.patch en un mismo directorio. Luego dentro del mismo descomprimir utilizando el comando

```
$ tarzvf OpenCaster2.4.tgz
```

Posteriormente se ejecuta el parche desarrollado por LIFIA

```
$ cd OpenCaster2.4/
$ patch -p1< ../OpenCaster2.4-lifia-rev362.patch
```

AVISO: El siguiente paso es suficiente para la mayoría de las referencias en las que se basa este manual, aún así para la fecha y prueba a la que se creó este documento este paso fue omitido y modificado debido a problemas con el uso de ciertos comandos del software (pesvideo2ts, tscbrmuxer y tsstamp).

Instalar OpenCaster con los privilegios del usuario root

```
$ sudo make
$ sudo makeinstall
```

Si se escoge probar hasta acá entonces continuar con siguiente sección, en caso contrario se debe descargar

Repositorio OpenCaster3.2.2+ParcheLIFIA-aplicado

Luego debemos copiar las carpetas pesvideo2ts, tscbrmuxer y tsstamp ubicadas en

`/OpenCaster2.4/tools`

para posteriormente reemplazar las carpetas con sus mismos nombres ubicadas en

`/opencaster_isdb-tb/tools`

el cual es el directorio por defecto del repositorio descargado.

Finalmente se instala OpenCaster-isdb-tb con los privilegios del usuario root

```
$ sudo make
$ sudo makeinstall
```

Para probar que la instalación fue realizada con exito, ejecutar el siguiente comando

```
$ python -c "from dvbobjects.PSI.PAT import *"
```

El comando debería ejecutar sin mostrar ningún error en consola.

3. Generación Transport Stream

Para crear un flujo único de paquetes de transporte TS inicialmente debemos escoger un video fuente, para este caso se descargó un video directamente de YouTube en formato AVI ya que a partir de las referencias en las que está basado este manual se comprobó que este formato compatibiliza correctamente con el programa OpenCaster.

El siguiente paso es obtener el Elementary Stream (ES) de audio

```
$ ffmpeg -i bruno_mars.avi -vn -ac 2 -acodec mp2 -f mp2 -ab 128000 -ar
48000 audio.mp2
```

Parámetro	**Descripción**
-i	Archivo fuente en formato .avi
-vn	Ignora la señal de audio.
-ac	Número de canales, por defecto 1.
-acodec	Códec de audio, se usa mp2 para compatibilidad con TS.
-f	Formato de salida.
-ab	Tasa de bits por segundo.
-ar	Frecuencia de muestreo de audio, por defecto es 44100Hz
audio.mp2	Nombre archivo de salida.

Tabla 1: Parámetros codificación de audio.

Luego el Elementary Stream (ES) de video

```
$ ffmpeg -i bruno_mars.avi -an -vcodec mpeg2video -f mpeg2video -s 1920x1080 -r 25
-aspect 16:9 -deinterlace -b 8000k -minrate 8000k -maxrate 8000k -bf 2 -bufsize
1835008 video.m2v
```

Parámetro	**Descripción**
-i	Archivo fuente en formato .avi
-an	Ignora la señal de audio.
-vcodec	Códec de video, se necesita que sea mpeg2video para compatibilidad con TS.
-f	Formato de salida.
-s	Resolución en píxeles.
-r	Frames por segundo.
-aspect	Relación de aspecto.
-deinterlace	Opción para desentrelazar imagenes.
-b	Tasa de bits en Kbps.
-bf	Num de cuadros tipo b para cada GOP.
-buffsize	Tamaño de búffer en bits.
video.m2v	Nombre archivo de salida.

Tabla 2: Parámetros codificación de video.

A continuación se encapsulan los ES en un Packetized Elementary Stream (PES) cada uno.

Primero el de video

```
$ esvideo2pes video.m2v > video.pes
```

Luego audio

```
$ esaudio2pes audio.mp2 1152 48000 384 3600 > audio.pes
```

Valor Parámetro	**Descripción**
1152	Número de muestras por frame.
48000	Frecuencia de muestreo, debe mantenerse la escogida en ES.
384	Tamaño de frame de audio.
3600	Desplazamiento del Presentation Time Stamp, se define para sincronizar video.
audio.mp2	Nombre archivo fuente a ser convertido.
audio.pes	Nombre archivo de salida convertido en formato PES.

Tabla 3: Parámetros esaudio2pes.

Luego se obtienen el archivo .ts de audio y de video

Para video

```
$ pesvideo2ts 2064 25 112 9200000 0 video.pes > video.ts
```

Valor parámetro	Descripción
2064	Número de PID video
25	Frames por segundo.
112	Valor de Video Buffer Verifier, en mpeg-2 está definido.
9200000	Ancho de banda de TS. Se recomianda que el ancho de banda sea un 15 % más grande que el bitrate escogido en función ffmpeg. Es decir 8000k*1.15=9200K.
0	Indica que el video no estará en loop.
video.pes	Nombre archivo fuente a ser convertido.
video.ts	Nombre archivo de salida convertido en formato TS.

Tabla 4: Parámetros pesvideo2ts.

Para audio

```
$ pesaudio2ts 2083 1152 48000 384 0 audio.pes > audio.ts
```

Valor Parámetro	Descripción
2083	Número de PID audio.
1152	Muestras por frame, mantener valor definido en PES.
48000	Frecuencia de muestreo, mantener valor definido en PES.
384	Tamaño de frame de audio, mantener valor definido en PES.
0	Indica que el audio no estará en loop.
audio.pes	Nombre archivo fuente a ser convertido.
audio.ts	Nombre archivo de salida convertido en formato TS.

Tabla 5: Parámetros pesaudio2ts.

Ya obtenido el archivo transport stream para audio y video, es necesario obtener las tablas SI/PSI. Para esto se muestra como a partir de un programa creado en python se pueden obtener las tablas SI/PSI y acomodar el contenido al archivo transport stream.

3.1. Obtención Tablas SI/PSI

Utilizando el editor de preferencia se debe crear el archivo gtables.py.

El encabezado debe llevar lo siguiente

```
1 #!/usr/bin/env python
2 import os
3 from dvbobjects.PSI.PAT import *
4 from dvbobjects.PSI.NIT import *
```

```
from dvbobjects.PSI.SDT import *
from dvbobjects.PSI.PMT import *
from dvbobjects.SBTVD.Descriptors import *
tvd_ts_id = 0x073b # ID de red.
tvd_orig_network_id = 0x073b # ID de red original.
ts_freq = 527.14 # Frecuencia de transmision
ts_remote_control_key = 0x05 # Tecla de control remoto.
tvd_service_id_sd = 0xe760 # ID de servicio de TV Digital.
tvd_pmt_pid_sd = 1031 # PID de la PMT del servicio.
```

Donde:

- tvd_ts_id es el identificador del Transport stream.
- tvd_orig_network_id es el identificador de red original.
- ts_freq es la frecuencia en que se transmite el Transport stream, en este caso 527.14 Mhz.
- ts_remote_control_key es la tecla de control remoto virtual, sirve para poder usar el control remoto para elegir el canal más rápido.
- tvd_service_id_sd es el identificador del servicio de TV digital.
- tvd_pmt_pid_sd es el PID que se usará para transmitir la información que componen el servicio.

3.1.1. Tabla NIT

A continuación se agrega el contenido para la tabla NIT.

```
nit = network_information_section(
    network_id = tvd_orig_network_id,
    network_descriptor_loop = [

        network_descriptor(network_name = "KLEINROCK_TV",),
        system_management_descriptor(
            broadcasting_flag = 0,
            broadcasting_identifier = 3,
            additional_broadcasting_identification = 0x01,
            additional_identification_bytes = [],
        )
    ],
    transport_stream_loop = [
        transport_stream_loop_item(
        transport_stream_id = tvd_ts_id,
        original_network_id = tvd_orig_network_id,
        transport_descriptor_loop = [
            service_list_descriptor(
                dvb_service_descriptor_loop = [
                    service_descriptor_loop_item (
                        service_ID = tvd_service_id_sd,
```

```
                        service_type = 1,
                    ),
                ],
            ),
            terrestrial_delivery_system_descriptor(
                area_code = 1341,
                guard_interval = 0x01,
                transmission_mode = 0x02,
                frequencies = [
                    tds_frequency_item( freq=ts_freq )
                ],
            ),
            partial_reception_descriptor (
                service_ids = []
            ),
            transport_stream_information_descriptor (
                remote_control_key_id = ts_remote_control_key,
                ts_name = "KLEINROCK_TV",
                transmission_type_loop = [
                    transmission_type_loop_item(
                        transmission_type_info = 0x0F,
                        service_id_loop = [
                            service_id_loop_item(
                                service_id=tvd_service_id_sd
                            ),
                        ]
                    ),
                    transmission_type_loop_item(
                        transmission_type_info = 0xAF,
                        service_id_loop = [],
                    ),
                ],
            )
        ],
        ),
    ],
    version_number = 0,
    section_number = 0,
    last_section_number = 0,
)
```

Donde:

- system_management_descriptor define propiedades del sistema, definiendo el sistema ISDB-Tb.
- terrestrial_delivery_system_descriptor define propiedades de la modulación, como intervalos de guarda, frecuencia de transmisión, etc.
- partial_reception_descriptor define la lista de servicios de recepción parcial. Esta lista tendría

que tener la lista de servicios 1-Seg.

- transport_stream_information_descriptor define otras propiedades del Transport stream que estamos creando, como la tecla de control remoto, el nombre del Transport stream, información de los tipos de servicios ofrecidos, etc.

3.1.2. Tabla SDT

Se agrega código para tablas SDT la cual especifíca los servicios disponibles en el Transport Stream.

```
sdt = service_description_section(
    transport_stream_id = tvd_ts_id,
    original_network_id = tvd_orig_network_id,
    service_loop = [
        service_loop_item(
            service_ID = tvd_service_id_sd,
            EIT_schedule_flag = 0,
            EIT_present_following_flag = 0,
            running_status = 4,
            free_CA_mode = 0,
            service_descriptor_loop = [
                service_descriptor(
                    service_type = 1,
                    service_provider_name = "",
                    service_name = "PRACTICA_22_23",
                ),
            ],
        ),
    ],
    version_number = 0,
    section_number = 0,
    last_section_number = 0,
)
```

3.1.3. Tabla PAT

Se agrega código para tabla PAT la cual tiene el mapa de servicios en PMTs. Las PMTs son tablas que definen como se compone cada servicio.

```
pat = program_association_section(
    transport_stream_id = tvd_ts_id,
    program_loop = [
        program_loop_item(
            # Programa especial para la tabla NIT
            program_number = 0,
            PID = 16,
        ),
        program_loop_item(
```

```
            program_number = tvd_service_id_sd,
            PID = tvd_pmt_pid_sd,
        ),
    ],
    version_number = 0,
    section_number = 0,
    last_section_number = 0,
)
```

3.1.4. Tabla PMT

Se agrega código para tabla PMT. Tiene que haber una PMT por cada servicio que se transmita. En este caso hay una sola. Esta define el flujo de datos que componen el servicio. En esta se detallan cual es el stream de audio, video, etc que componen cada servicio y además se define en que PID se encuentra el PCR que es la marca del reloj del servicio.

```
pmt_sd = program_map_section(
        program_number = tvd_service_id_sd,
        PCR_PID = 2064,
        program_info_descriptor_loop = [],
        stream_loop = [
                stream_loop_item(
                        stream_type = 2, # mpeg2 video stream type
                        elementary_PID = 2064,
                        element_info_descriptor_loop = []
                ),
                stream_loop_item(
                        stream_type = 3, # mpeg2 audio stream type
                        elementary_PID = 2083,
                        element_info_descriptor_loop = []
                ),
        ],
        version_number = 0,
        section_number = 0,
        last_section_number = 0,
)
```

3.1.5. Tablas a archivos

A continuación se muestra el código necesario para escribir las tablas anteriores en archivos.

```
out = open("./nit.sec", "wb")
out.write(nit.pack())
out.close
os.system("sec2ts 16 < ./nit.sec > ./nit.ts")
```

```
7  out = open("./pat.sec", "wb")
8  out.write(pat.pack())
9  out.close
10 os.system("sec2ts 0 < ./pat.sec > ./pat.ts")
11
12
13 out = open("./sdt.sec", "wb")
14 out.write(sdt.pack())
15 out.close
16 os.system("sec2ts 17 < ./sdt.sec > ./sdt.ts")
17
18
19 out = open("./pmt_sd.sec", "wb")
20 out.write(pmt_sd.pack())
21 out.close
22 os.system("sec2ts " + str(tvd_pmt_pid_sd) +
23         " < ./pmt_sd.sec > ./pmt_sd.ts")
```

3.1.6. Ejecutar gtables.py

Ya completado el archivo, este debe ser ejecutado mediante el siguiente comando:

```
$ chmod u+x gtables.py
$ ./gtables.py ó $ python gtables.py
```

3.2. Finalización archivo transport stream

Si los pasos fueron seguidos correctamente ahora se deberían tener los archivos **pat.s, pmt_sd.ts, sdt.ts y nit.ts.**

A continuación hace falta obtener un archivo nulo llamado null.ts, este se puede encontrar en la dirección

```
$ /OpenCaster2.4/tutorials/OCTutorial2
```

Llegado a este punto **se deben tener los siguientes archivos** y además estar en el mismo directorio:

- video.ts
- audio.ts
- pat.ts
- pmt_sd.ts
- sdt.ts
- nit.ts
- null.ts

Se procede a multiplexar los archivos mediante el comando

```
$ tscbrmuxer 4960000 b:15040 pat.ts b:15040 pmt_sd.ts b:3008 sdt.ts b:3008
nit.ts b:9200000 video.ts b:188000 audio.ts b:20534198 null.ts > bruno_prueba.ts
```

Valor Parámetro	Descripción
4960000	Paquetes a multiplexar. ISDB-Tb transmite cerca de 20000 paquetes por segundo. El video escogido dura 04:08min, por lo tanto se van a necesitar 4960000 paquetes.
15040	PMT y PAT se deben enviar al menos 10 veces por segundo. Si cada una entra en un solo paquete de 188 bytes, entonces 188 bytes * 8 * 10 segundos = 15040 bits.
3008	SDT y NIT sigue el mismo análisis de PMT y PAT, solo que estas son enviadas cada 2 segundos.
9200000	Ancho de banda de video que ya se definió previamente.
188000	Ancho de banda de audio que ya se definió previamente.
20534198	Ancho de banda para los paquetes nulos. La norma ISDB-Tb tiene un ancho de banda fijo de 29.958.294 bps. Por lo tanto para determinar este valor se realiza el siguiente cálculo: 29958294 - 15040*2 - 3008*2 - 9200000 - 188000 = 20534198
bruno_prueba.ts	Nombre archivo de salida multiplexado.

Tabla 6: Parámetros tscbrmuxer

Al realizar la multiplexación se desincronizan los paquetes de video, por lo que su llegada al Set Top-Box será imprecisa. Para solucionar esto se utiliza el comando tsstamp para sincronizar el audio con el video.

```
$ tsstamp bruno_prueba.ts 29958294 > bruno_prueba.fixed.ts
```

Así finalmente se obtiene el archivo transport stream listo para ser transmitido.

4. Transmisión Transport Stream

El proceso de modulación y transmisión del archivo transport stream se raliza con la tarjeta moduladora DTA-115, hardware desarrollado por DekTec, con su respectivo software para Linux DtPlay.

Para esto se debe mover el archivo bruno_prueba.fixed.ts a la carpeta DtPlay y ejecutar el comando

```
$ ./DtPlay bruno_prueba.fixed.ts -t 115 -i 2 -mf 527.143 -m 188 -r 31457280
-mt ISDBT -l 0
```

Parámetro	Descripción
-t	Indica el tipo de tarjeta DekTec que se va a ocupar.
-i	Indica el puerto de salida de la tarjeta moduladora.
-mf	Indica la radiofrecuencia en donde se va a transmitir.
-m	Indica el modo de transmisión, en este caso 188 bytes, quieredecir que el transport stream se encuentra en paquetes de 188 bytes, y transmite de la misma manera.
-r	Indica el Transport Stream rate en bps que se desea transmitir.
-mt	Indica la norma que se va a ocupar para transmitir, en este caso es ISDB-Tb.

Tabla 7: Parámetros Transmisión con DtPlay.

5. Resultados

La figura 1 muestra como se ve el comando de DtPlay al momento de ser ejecutado.

```
[ctronc:...sktop/Practica_22_23/DtPlay]$ ./DtPlay bruno_prueba.fixed.ts -t 115 -i 2 -mf 52
7.143 -m 188 -r 31457280 -mt ISDBT
DtPlay player V4.6.2 (c) 2000-2015 DekTec Digital Video B.V.

DTAPI compile version: V5.16.2.71

DTAPI link version: V5.16.2.71

ACA_IsdbtPars

Start playing
- Play file name        : bruno_prueba.fixed.ts
- Loop file             : 1x
- Transmit Mode         : 188
- Output device         : DTA-115 port 2 (#1)
- Modulation Type       : ISDB-T
- Carrier Frequency     : 527.14 MHz
- Output Level          : -27.5 dBm

Press any key to stop playing
```

Figura 1: Comando DtPlay ejecutado.

La figura 2 muestra la sintonización de la señal, se observa como en 527MHz se logra recibir el video tal como se asignó previamente.

Figura 2: Sintonizando señal.

La figura 3 muestra como el video se está transmitiendo en la televisión de forma satisfactoria.

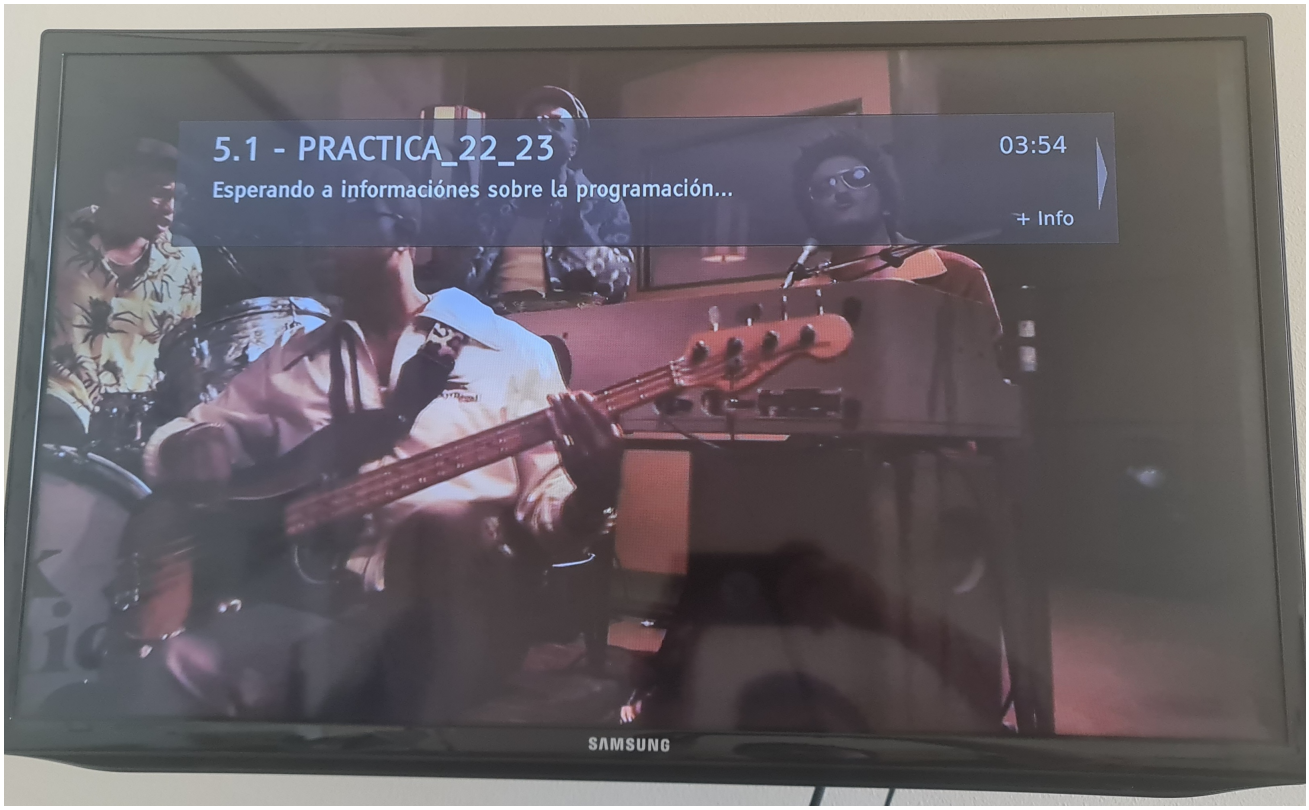

Figura 3: Transmitiendo video.

6. Conclusión

En este manual se realizó una guía paso a paso para transmitir un video mediante la tarjeta moduladora DTA-115 a la televisión digital de manera satisfactoria.

- Se expusieron los requisitos para realizar el experimento.
- Se explicó cómo instalar software específico.
- Se explicó cómo crear un archivo transport stream.
- Se explicó cómo transmitir con el software de la tarjeta moduladora.

7. Referencias

- Moncayo T., Pozo M., Mejía D. y Bernal I. Generador de Flujos Únicos de Paquetes de Transporte TS en base a la Norma ISDB-Tb. Fuente.
- AVALPA Broadcast Server User Manual. Fuente.
- LIFIA. OpenCaster para SATVD-T Fuente.
- Diego Villamarín, Gonzalo Olmedo, Román Lara y María Augusta Illescas. Generador de Transport Stream con Audio, Video y Datos de interactividad para el Sistema de Televisión Digital Terrestre ISDB-Tb Fuente.
- Marcelo Alejandro Gutiérrez Vidal. Sistema de difusión de alertas de emergencia bajo norma ISDB-T.
- María de los Ángeles Mateos Moreno. Generación de un Transport Stream de televisión interactiva. Fuente.