ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"IMPLEMENTACIÓN DE UN PROTOTIPO DE GRABACIÓN AUTOMATIZADO DE SEÑAL DE TELEVISIÓN ABIERTA"

TESIS DE GRADO

Previa a la obtención del título de

INGENIERO EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

Presentado por:

DIEGO GUSTAVO CAIZA MÉNDEZ

JUAN JOSÉ PÉREZ INSUASTI

RIOBAMBA – ECUADOR
2011

DEDICATORIA

Esta tesis va dedicada con mucho afecto a todos aquellos que hicieron posible culminar esta etapa de mi vida. En especial a mi abuelita quien ha estado siempre presente brindándome su constante apoyo y confianza, gracias al gran esfuerzo que ha realizado para darme la educación que hoy me ha permitido llegar hasta aquí. A mi abuelito por su incondicional ayuda y cariño. A mi mamá que con sus cuidados y atenciones me ha apoyado a ver alcanzada mi meta. A toda mi familia que siempre supieron darme una mano de soporte y nunca me dejaron desmayar frente a las adversidades que se me presentaron.

Diego

DEDICATORIA

A mi padre que me enseño que las ideas se pueden convertir en realidades. A mi familia quienes siempre me alentaron a seguir adelante"

Juan José

NOMBRE	FIRMA	FECHA
Ing. Iván Menes **DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Pedro Infante **DIR.ESC.ING.ELECTRÓNICA EN TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Santiago Cisneros **DIRECTOR DE TESIS**	_____	_____
Ing. Edwin Altamirano **MIEMBRO DEL TRIBUNAL**	_____	_____
Lcdo. Carlos Rodríguez **DIR. DPTO. DOCUMENTACIÓN**	_____	_____

NOTA DE LA TESIS _____

“Nosotros, Juan José Pérez Insuasti y Diego Gustavo Caiza Méndez somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”

_____	_____
Juan José Pérez Insuasti	Diego Gustavo Caiza Méndez

ÍNDICE DE ABREVIATURAS

ASF	Advanced Systems Format
AVI	Audio Video Interleave
BBC	British Broadcasting Corporation
COM	Component Object Model
DLL	Dynamic Link Library
DV	Digital Video
DVB-C	Digital Video Broadcasting - Cable
DVB-T	Digital Video Broadcasting - Terrestrial
DVD	Digital Versatile Disc
DVI	Digital Visual Interface
FPS	Frames por Segundo
GOP	Group of Pictures
GUID	Globally Unique Identifier
HDMI	High-Definition Multimedia Interface
HF	Hugh Frecuency
MF	Medium Frecuency
MPEG	Motion Picture Experts Group
NBC	National Broadcasting Company
NTSC	National Television System Commission
PAL	Phase Alternation Line

RCA	Radio Corporation of America
RGB	Red Green Blue
SECAM	Sequential Color with Memory
TDT	Televisión digital terrestre
UHF	Ultra High Frecuency
UIT	Unión Internacional de Telecomunicaciones
VGA	Video Graphics Array
VHF	Very High Frecuency
WDM	Windows Driver Model
WMA	Windows Media Audio

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La televisión es un medio de información y entretenimiento para sus espectadores. Alrededor de un 96% de hogares ecuatorianos posee por lo menos un televisor, según estudios realizados por el Instituto Nacional de Estadísticas y Censos, en el 2010.

La televisión es el medio de comunicación más utilizado en cuanto a publicidad se refiere. Un estudio realizado por la empresa Infomedia concluye que la TV pautó el 59% de publicidad en el 2010. Los medios impresos ocuparon el segundo lugar con el 26% y la radio con el 6%. Los principales anunciantes fueron los estamentos públicos, seguidos estrechamente por las empresas de telefonía móvil.

Siendo la televisión el medio más utilizado por los anunciantes, las agencias de publicidad han tenido que redoblar esfuerzos para monitorear el cumplimiento de pautas contratadas. Mientras que los usuarios de la televisión buscan alternativas para grabar programas de su interés que no los pueden ver en directo. Además, los organismos de control del estado como la SUPERTEL requieren realizar monitoreos y/o grabaciones de las señales televisivas con diferentes finalidades.

El monitoreo y control de medios televisivos demanda emplear recursos tanto humanos como de infraestructura técnica. Las empresas que se dedican a esta labor, requieren contratar cada vez un mayor número de personas, para realizar los monitoreos y grabaciones.

La grabación bajo el sistema analógico era lenta, de baja calidad, y además ocupaba muchísimo espacio. El paso al sistema digital aporta infinidad de ventajas,

superando ampliamente en prestaciones a los clásicos sistemas de grabación analógica. La tendencia actual es dar soporte para la grabación digital de la información.

El sistema planteado en este proyecto, básicamente está diseñado para ahorrar recursos. Los administradores pueden monitorear y grabar programas o pautas, sin la necesidad de contar con una costosa infraestructura, ni demandar mucha mano de obra.

El software es implementado bajo la plataforma Visual Studio.NET, a través del lenguaje C#, permite el monitoreo de los canales de televisión, así como las funciones de grabación en distintas modalidades para día y hora que se programe en forma autónoma. El respaldo de la información se almacena en sistemas de archivos.

Este sistema será de utilidad tanto para los organismos de control del Estado, así como para empresas que anuncian sus productos en la televisión, agencias de publicidad. En general puede ser utilizado por entidades y personas que necesitan monitorear las emisiones que se transmiten a través de los medios televisivos. También puede ser utilizado por los consumidores de la televisión que requieren ver programas que salen al aire en tiempos en los que no pueden verlos en directo.

CAPÍTULO I

INTRODUCCIÓN

1.1 ORIGEN Y DESARROLLO DE LA TELEVISIÓN

La televisión es un sistema para la transmisión y recepción de imágenes en movimiento y sonido a distancia. Esta transmisión es efectuada mediante la emisión de ondas electromagnéticas. El receptor de las señales es el televisor.

Las primeras emisiones de prueba fueron realizadas en Inglaterra en 1927 y estuvieron a cargo de la BBC, lo mismo ocurrió en 1930 en Estados Unidos bajo la responsabilidad de la NBC y la CBS.

A partir de 1941 se estandariza el formato para televisión en blanco y negro de 325 líneas, para todo el territorio estadounidense. Terminada la guerra en 1945, la industria de la Televisión se introduce en la sociedad norteamericana, vendiéndose más de 10 millones de receptores. Las pruebas realizadas por la CBS y la RCA, sirvieron para crear la NTSC (National Television System Commission) con el formato de 525 líneas y 6Mhz de ancho de banda. En 1953 Estados Unidos se convierte en el primer país en transmitir Televisión a Color y el sistema creado es adoptado como estándar.

Años más tarde, en 1963, en Alemania, los laboratorios Telefunken desarrollan el sistema PAL (Phase Alternation Line) como una mejora de NTSC y que es implantado en la mayor parte del territorio europeo.

Por consideraciones de carácter político, Francia crea su propio estándar denominado SECAM (Sequential Couleur Avec Memoire) y al igual que PAL consta de 625 líneas.

Los tres sistemas nombrados son los que paulatinamente se tomaron el mundo para brindar el servicio de televisión analógica.

1.2 ESTÁNDARES DE TELEVISIÓN ANALÓGICA

Los sistemas de televisión analógica son diferentes en Europa y en América. La falta de acuerdos y la carrera por desarrollar un estándar superior al de la competencia, no permitieron la creación de una norma universal para la televisión analógica, pues los sistemas son incompatibles entre sí.

1.2.1 NTSC

NTSC es la sigla de National Television System Commission, en español Comisión Nacional de Sistemas de Televisión, es un sistema de codificación y transmisión de Televisión a color analógica desarrollada en Estados Unidos en los años 40.

Primero se trabajó con un sistema de 325 líneas, utilizando un ancho de banda de 6 MHz, garantizando la compatibilidad entre sistemas B/N (blanco y negro) y a color.

En 1953 el estándar es modificado dando paso al actual formato que es conocido como: NTSC-M en el cual se usa 525 líneas por cuadro y adicionalmente quedan definidas las siguientes especificaciones: frecuencia de campo (frecuencia de barrido vertical) de 60 Hz, frecuencia de barrido horizontal de 15.750 Hz, relación de aspecto 4:3, ancho de banda de 6 MHz, en el cual se incluye la portadora de video y de audio.

El estándar NTSC-M es usado mayoritariamente en América del Norte, América del Sur, Centro América y Japón. Cuando se introdujo la televisión en el Ecuador, se optó por el estándar NTSC-M, con las condiciones técnicas descritas anteriormente.

SISTEMA NTSC-M	
Líneas / Campos	525 / 60
Frecuencia Barrido Horizontal	15.734 KHz
Frecuencia Barrido Vertical	60 Hz
Frecuencia Subportadora de color	3.579545 MHz
Ancho de Banda	6 MHz
Portadora de Sonido	4.5 MHz

Tabla I. I Características de NTSC-M

1.2.2 PAL

PAL es la sigla de Phase Alternating Line, en español Línea Alternada en Fase. Es de origen alemán y se utiliza en la mayoría de los países africanos, asiáticos y europeos, además de Australia y algunos países latinoamericanos.

El sistema PAL surgió en el año 1963, con el fin de mejorar la calidad y reducir los defectos en los tonos de color que presentaba el sistema NTSC. No obstante, los conceptos fundamentales de la transmisión de señales han sido adoptados del sistema NTSC.

El sistema de color PAL se usa habitualmente con un formato de vídeo de 525 líneas por cuadro y una tasa de refresco de pantalla de 25 cuadros por segundo, entrelazadas, como ocurre por ejemplo en las variantes PAL-B, G, H, I y N. Algunos países del Este de Europa que abandonaron el sistema SECAM ahora emplean PAL D o K, adaptaciones para mantener algunos aspectos técnicos de SECAM en PAL.

El sistema PAL es más robusto que el sistema NTSC. En cualquier caso en el que haya rebotes de señal, el sistema PAL se ha demostrado netamente superior al NTSC. Esa fue la razón por la que la mayoría de los países europeos eligieron el sistema PAL, ya que la geografía europea es mucho más compleja que la norteamericana. Otro motivo es que en los EE.UU. son habituales las emisiones de carácter local y en Europa lo son las estaciones nacionales, cuyas emisoras suelen tener un área de cobertura más extensa. En el único aspecto en el que el NTSC es superior al PAL es en evitar la sensación de parpadeo que se puede apreciar en la zona de visión periférica cuando se mira la TV en una pantalla grande (más de 21

pulgadas), porque la velocidad de refresco es superior (30Hz en NTSC frente a 25Hz en PAL).

Comúnmente en algunos países de Latinoamérica, fabricantes de equipos de vídeo presentan receptores trinorma que pueden ser usados en cualquier país del continente americano, donde NTSC-M, PAL-M y PAL-N son las normas usadas.

SISTEMA	PAL B, G, H	PAL I	PAL D	PAL N	PAL M
Línea/Campos	625/50	625/50	625/50	625/50	525/60
Frecuencia Horizontal	15.625 Khz	15.625 Khz	15.625 Khz	15.625 Khz	15.750 Khz
Frecuencia Vertical	50 Hz	50 Hz	50 Hz	50 Hz	50 Hz
Frecuencia Subportadora de Color	4.4336 MHz	4.4336 MHz	4.4336 MHz	3.5820 MHz	3.5756 MHz
Ancho de Banda de video	5.0 MHz	5.5 MHz	6.0 MHz	4.2 MHz	4.2 MHz
Portadora de Sonido	5.5 MHz	6.0 MHz	6.5 MHz	4.5 MHz	4.5 MHz

Tabla I. II Características de PAL

1.2.3 SECAM

Secam son las siglas de Sequential Couleur Avec Memoire, en español Color Secuencial con Memoria. El sistema SECAM fue inventado por un equipo liderado por Henri de France trabajando para la firma Thompson. Se basa en la transmisión de una señal monocromática que transmite la luminancia complementada con una segunda señal que porta la información del color.

Con propósitos de compatibilidad, no usa más ancho de banda que el que ocupa la señal monocroma. La segunda señal es insertada en la primera pero sin interferirla. En idea es posible, ya que la señal monocromática no es continua y deja espacios vacíos que pueden ser utilizados. Los sistemas de TV en color analógicos se diferencian en la forma en que se usan esos espacios libres. La señal de color, en todos los casos, se inserta al final del espectro de la señal monocroma.

SISTEMA	SECAM B, G, H	SECAM D, K, K1, L
Línea/Campos	625/50	625/50
Frecuencia Horizontal	15.625 KHz	15.625 KHz
Frecuencia Vertical	50 Hz	50 Hz
Ancho Banda Video	5.0 MHz	6.0 MHz
Portadora Sonido	5.5 MHz	6.5 MHz

Tabla I. III Características de SECAM

1.3 TELEVISIÓN ANALÓGICA EN EL ECUADOR

En nuestro país la radiodifusión televisiva aparece a mediados de los años 60. Actualmente, la señal de televisión llega a la mayor parte del territorio nacional. Están al aire emisoras televisivas de alcance local, regional o de cobertura nacional.

1.3.1 Información General

Dentro de la Ley de Radiodifusión y Televisión del Ecuador se considera que para el Estado Ecuatoriano los Canales y Frecuencias Radioeléctricas constituyen un "Patrimonio Nacional", por lo que el Estado es el responsable de su control, regulación y concesión. En esta Ley se expresa que la Televisión representa: "La

comunicación visual y sonora unilateral a través de la emisión de ondas electromagnéticas para ser visualizadas y escuchadas por el público en general".

Dentro de todo el país el sistema establecido es NTSC-M de 525 líneas de acuerdo a las características técnicas que se establecen por la UIT y complementariamente por la FCC. Haciendo referencia a la Norma Técnica para el Servicio de Televisión Analógica y Plan de Distribución de Canales, publicada en el Registro Oficial Número 335 del 29 de mayo del 2001.

1.3.2 El Espectro Radioeléctrico Para Radiodifusión

La UIT ha dividido al mundo en tres regiones, Ecuador se encuentra en la Región II. Las bandas se han designado en orden creciente. Para radiodifusión AM, FM y TV son las siguientes:

Rango de Frecuencias	Designación	Usos
300 KHz a 3 MHz	MF (Medium Frecuency)	Radio AM
3 MHz a 30 MHz	HF (High Frecuency)	
30 MHz a 300 MHz	VHF (Very High Frecuency)	Radio FM y TV
300 MHz a 3 GHz	UHF (Ultra High Frecuency)	TV

Tabla I. IV El espectro radioeléctrico para radiodifusión

La banda designada para la señal de televisión va de 54 MHz a 890 MHz. Cada canal ocupa 6 MHz. Por ello, hay espacio para que operen 83 canales.

1.3.3 Bandas de Operación y Distribución de Canales

El espectro radioeléctrico se divide de acuerdo al Reglamento de Radiocomunicación de la Unión Internacinal de Telecomunicaciones (ITU). En la

Norma Técnica para Televisión y de acuerdo al Plan Nacional de Distribución de Frecuencias se establece que las bandas de VHF y UHF para televisión abierta son:

BANDA	FRECUENCIA	CANALES
Banda I	54 a 72 MHz	2 al 4
	76 a 88 MHz	5 al 6
Banda III	174 a 216 MHz	7 al 13
Banda IV	500 a 608 MHz	19 al 36
	614 a 644 MHz	38 al 42
Banda V	644 a 686 MHz	43 al 49

Tabla I. V Banda de Frecuencias para TV abierta en VHF u UHF

Los canales 19 y 20 se encuentran reservados, a fin de facilitar el proceso de transición hacia la Televisión Digital. Los canales 48 y 49 en UHF se hallan reservados para el Estado Ecuatoriano.

Usando NTSC-M hay espacio para 42 canales, cada uno con un ancho de banda de 6 MHz y está establecido en la Norma Técnica de la siguiente manera:

RANGO DE FRECUENCIAS	BANDA	CANAL	FRECUENCIA	PORTADORAS	
MHz		Nro.	MHz	Vídeo MHz	Sonido MHz
VHF 54 - 72	I	2	54-60	55.25	59.75
		3	60-66	61.25	65.75
		4	66-72	67.25	71.75
VHF 76 - 88	I	5	76-82	77.25	81.75
		6	82-88	83.25	87.75
VHF 174 - 216	III	7	174-180	175.25	179.75
		8	180-186	181.25	185.75
		9	186-192	187.25	191.75
		10	192-198	193.25	197.75

		11	198-204	199.25	203.75
		12	204-210	205.25	209.75
		13	210-216	211.25	215.75
UHF 500 – 608	IV	19	500-506	501.25	505.75
		20	506-512	507.25	511.75
		21	512-518	513.25	517.75
		22	518-524	519.25	523.75
		23	524-530	525.25	529.75
		24	530-536	531.25	535.75
		25	536-542	537.25	541.75
		26	542-548	543.25	547.75
		27	548-554	549.25	553.75
		28	554-560	555.25	559.75
		29	560-566	561.25	565.75
		30	566-572	567.25	571.75
		31	572-578	573.25	577.75
		32	578-584	579.25	583.75
		33	584-590	585.25	589.75
		34	590-596	591.25	595.75
		35	596-602	597.25	601.75
		36	602-608	603.25	607.75
UHF 614 – 644	IV	38	612-620	615.25	619.75
		39	620-626	621.25	625.75
		40	626-632	627.25	631.75
		41	632-638	633.25	637.75
		42	638-644	639.25	643.75

UHF 644-686	V	43	644-650	645.25	649.75
		44	650-656	651.25	655.75
		45	656-662	657.25	661.75
		46	662-668	663.25	667.75
		47	668-674	669.25	673.75
		48	674-680	675.25	679.75
		49	680-686	681.25	685.75

Tabla I. VI Distribución de Canales de VHF – UHF con ancho de banda de 6 MHz

El canal 37 (608-614 MHz) está destinado al servicio de Radioastronomía.

1.3.4 Distribución Geográfica Para Televisión Abierta en VHF y UHF

La asignación de canales de televisión abierta se la realiza de acuerdo a las especificaciones detalladas en la Norma Técnica para Televisión y de acuerdo al Plan de Distribución de Canales. Esta división por zonas ha sido realizada con el fin de proporcionar grupos de canales que estarán libres de interferencias por canal adyacente.

La Tabla I. VII expresa la división por zonas geográficas y el grupo de canales disponibles para cada zona tanto en las bandas de VHF como UHF.

DESCRIPCIÓN DE LA ZONA GEOGRÁFICA (Norma Técnica)	GRUPO	GRUPO
ZONA A: Azuay excepto zona norte (cantones Sigsig, Chordeleg, Gualaceo, Paute, Guachapala, El Pan y Sevilla de Oro)	A1,B2	G1,G4
ZONA B: Bolívar y Chimborazo, excepto cantón Echeandía y zona occidental de la Cordillera Occidental.	A1,B2	G1,G4
ZONA C: Provincia del Carchi.	A1,B2	G1,G4
ZONAD:Provincias de Orellana y Sucumbíos	A1,B2	G1,G4

ZONA E: Provincia de Esmeraldas, excepto Rosa Zárate y Muisne	A1,B2	G1,G3
ZONA G1: Guayas, subzona1: excepto Santa Elena, Gral. Villamil, El Empalme, Palestina y Balao, se incluye La Troncal, Suscal y zona occidental de la Cordillera	A1,B1	G2,G4
ZONA G2: Guayas, subzona 2: Santa Elena y Gral. Villamil	A1,B2	G1,G3
ZONA J: Provincia de Imbabura.	A1,B2	G1,G3
ZONA L1: Loja, excepto cantones de Loja, Catamayo, Saraguro, Amaluza y zona occidental de la Cordillera Occidental.	A2,B1	G2,G3
ZONA L2: Provincia de Loja: cantones Loja, Catamayo y Saraguro.	A1,B2	G2,G3
ZONA M1: Provincia de Manabí, zona norte (desde Ricaurte al norte), excepto El Carmen y Flavio Alfaro; se incluye Muisne.	A2,B1	G2,G4
ZONA M2: Provincia de Manabí, zona sur, desde San Vicente al sur, excepto Pichincha	A1,B2	G2,G3
ZONA N: Provincia de Napo	A1,B2	G2,G4
ZONA Ñ: Provincia de Cañar, excepto zona occidental Cordillera Occidental (Suscal, La Troncal) e incluye zona norte provincia de Azuay	A2,B1	G2,G3
ZONA O: Provincia de El Oro y zona occidental de la Cordillera	A1,B2	G1,G3
ZONA P1: Pichincha, excepto zona occidental de A1, B1 G2, G4 (Santo Domingo y Los Bancos, P.V. Maldonado.	A1,B2	G2,G4
ZONA P2: Pichincha, zona de Santo Domingo, A2,B2 G1,G3	A2,B2	G1,G3
ZONA R1:Los Ríos, excepto Quevedo, Buena Fe, A1,B2 G2,G4	A1,B2	G2,G4
ZONA R2: Provincia de Los Ríos, Quevedo, Buena Fe, A2,B2 G1,G3	A1,B2	G1,G3
ZONA S1: Morona Santiago, excepto cantón Gral. A2, B2 G2, G4 Plaza al sur.	A1,B2	G1,G3
ZONA S2: Provincia de Morona Santiago, cantón Gral. Plaza A1, B2 G2, G4 al sur.	A1,B2	G1,G3
ZONA T: Provincias de Tungurahua y Cotopaxi, excepto zona A1, B1 G2, G3 occidental de la Cordillera Occidental.	A1,B2	G1,G3
ZONA X: Provincia de Pastaza. A1, B2, G1, G3	A1,B2	G1,G3
ZONA Y: Provincia de Galápagos. A1, B2	A1,B2	G1,G3

ZONA Z: Provincia de Zamora Chinchipe, incluye cantón A1, B2 G1, G3 Amaluza	A1,B2	G1,G3

Tabla I. VII Zonas Geográficas Para Televisión Abierta VHF Y UHF

La distribución geográfica por zonas a las que hace referencia la Norma Técnica para Televisión en Ecuador puede ser visualizado en el siguiente mapa.

Figura I. 1 Distribución Geográfica para el funcionamiento de TV analógica

CAPÍTULO II

FUNDAMENTOS DEL VÍDEO ANALÓGICO Y DIGITAL

2.1 INTRODUCCIÓN

El video es la tecnología de la captación, grabación, procesamiento, almacenamiento, transmisión y reconstrucción por medios electrónicos digitales o analógicos de una secuencia de imágenes que representan escenas en movimiento.

Es un proceso electrónico que puede ser analógico o digital. La sucesión de imágenes fijas crea la sensación de movimiento que capta, almacena y reproduce el vídeo.

La señal de vídeo analógica se produce tras convertir los cambios de la intensidad de la luz en señales eléctricas. Estas señales son impresas en materiales fotosensibles como las cintas de vídeo.

Las imágenes fijas del vídeo son conocidas como frames. La frecuencia a la que son reproducidas estas imágenes se denomina framerate y viene dado en frames por segundo (fps).

2.2 VIDEO ANALÓGICO

La señal de vídeo analógico se consigue a través del muestreo periódico de la información capturada que llega a la pantalla. Este proceso es conocido como barrido o scanning. A través de él se obtienen los datos de luminancia y crominancia.

La luminancia es la señal que dará la información sobre la intensidad de la luz. La señal de crominancia es el componente de la señal de vídeo que contiene las informaciones del color, comprende los canales rojo, verde y azul. El sistema de vídeo compuesto es el que tiene ambas señales.

En vídeo analógico la calidad de la señal dependerá de la calidad del grabador, del soporte y del reproductor. Esta calidad se reducirá en función del número de veces que sea reproducido.

Los diferentes soportes de vídeo analógico son muy similares a los de audio analógico. Son cintas electromagnéticas protegidas por una carcasa plástica.

El soporte de vídeo analógico más popular es la cinta VHS. También han existido soportes como el Betamax, el U-Matic, el Betacam o el Super 8. La mayor parte de ellos están en desuso.

2.3 VIDEO DIGITAL

Es una representación digital, la información se representa en forma de bits de la señal analógica. La calidad de reproducción de un sistema digital de vídeo bien diseñado es independiente del medio y depende únicamente de la calidad de los procesos de conversión.

Existe una serie de diferencias y ventajas notables entre el vídeo digital y analógico, entre ellas se destacan las siguientes:

- En el sistema de vídeo digital la calidad de imagen se encuentra totalmente independiente, solamente se ve afectada durante el proceso de digitalización de la misma. En el sistema analógico depende de la calidad de la cinta de vídeo, el reproductor, etc.
- El sistema digital permite la edición no lineal, se pueden editar las imágenes y el sonido de forma más rápida sin seguir ningún orden. En cambio, en el sistema de video analógico la edición sigue el criterio lineal, ya que depende de un soporte de cinta magnética y exige seguir un orden en la secuencia de filmación.
- Los colores en la edición de vídeo analógico se encuentran limitados a la hora de reproducir una imagen en los niveles de contraste y brillo. El video

digital utiliza los tres colores primarios, haciendo que los colores de la imagen se definan de forma más exacta.

- En el sistema de vídeo digital la realización de copias no presentan ninguna clase de perdida en calidad, pudiéndose realizar tantas copias como se desee. En cambio en el sistema analógico la calidad depende del número de copias y grabaciones además de otros componentes externos.

2.3.1 Digitalización de la señal de video

Digitalizar, quiere decir convertir el vídeo o señal analógica a lenguaje máquina.

El proceso de digitalización se lo realiza en tres pasos:

- Muestreo
- Cuantificación y
- Codificación

Muestreo: El muestreo digital es uno de los procesos que permite la digitalización de las señales. Consisteen tomar muestras periódicas de la amplitud de la señal analógica. Estas muestras (samples) no se toman de forma aleatoria o al azar, sino que se toman en intervalos fijos de tiempo (de ahí que hayan sido definidas como periódicas).

Cada muestra debe durar el mismo tiempo y efectuarse en el mismo intervalo. La velocidad a la que se hace este muestreo, es decir, el número de muestras que se toman por segundo es lo que se conoce como frecuencia de muestreo.

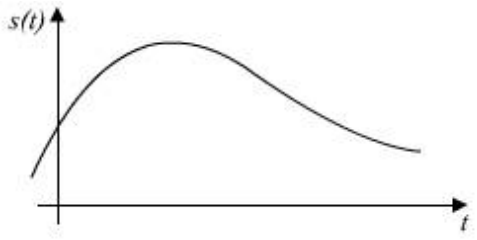


Figura II. 2 Señal analógica antes del muestreo

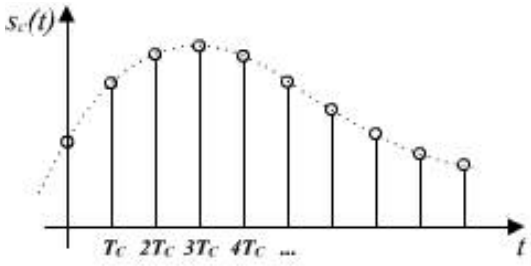


Figura II. 3 Señal digital después del muestreo

- **Frecuencia de muestreo:** La tasa o frecuencia de muestreo es el número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, durante el proceso necesario para convertirla de analógica en digital.
- **Teorema de Nyquist:** Según el teorema de muestreo de Nyquist-Shannon, para poder replicar con exactitud la forma de una onda es necesario que la frecuencia de muestreo sea como mínimo el doble de la máxima frecuencia a muestrear.

  El teorema de Nyquist sólo indica el valor mínimo necesario para que el muestreo resulte eficaz. Por encima de ese valor, cuanto mayor sea el número de niveles de comparación (muestras), más fiel será la conversión analógica digital (A/D), lo que se traduce en una mayor calidad de la señal

resultante. Cuantas más muestras se tengan, será posible reconstruir mejor la señal; no obstante, a mayor frecuencia de muestreo (más información/datos), mayor será el ancho de banda necesario.

- **Aliasing:** Este razonamiento fue deducido por Nyquist-Shannon, al establecer que para conseguir un muestreo-recuperación sin distorsión, se requiere que la frecuencia de muestreo f_o sea al menos dos veces más elevada que la frecuencia máxima presente en la señal analógica muestreada.

  La recuperación de la banda base se realizaría con un filtro pasa bajo que corte todas las frecuencias superiores $f_o/2$. De no cumplirse el teorema del muestreo de Nyquist, el filtro dejaría pasar frecuencias pertenecientes a la banda lateral inferior contaminantes de la banda base, que producirían solapamientos con las frecuencias más altas de la misma. Este efecto se denomina "aliasing" (Figura II. 4).

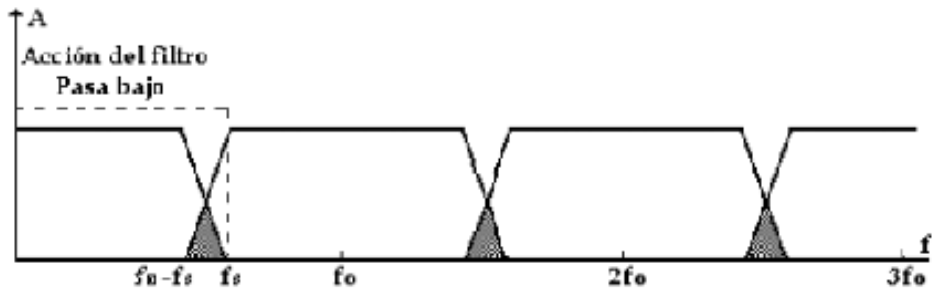


Figura II. 4 Cuando la frecuencia de muestreo es fo < 2fs

Cuantificación: La cuantificación lo que hace es convertir una sucesión de muestras de amplitud continua en una sucesión de valores de amplitudes discretas, o lo que es lo mismo, en una señal digital, aunque no binaria.

Durante el proceso de cuantificación, se mide el nivel de voltaje de cada una de las muestras, obtenidas en el proceso de muestreo, y se les atribuye a un valor determinado (discreto) de amplitud, seleccionado por aproximación dentro de un margen de niveles previamente fijado.

Los valores preestablecidos para ajustar la cuantificación se eligen en función de la propia resolución que utilice el código empleado durante la codificación. Si el nivel obtenido o medido de la señal original no coincide exactamente con ninguno de los valores preestablecidos por la cuantificación, se toma como valor el inferior más próximo. La señal analógica (que puede tomar cualquier valor) se convierte en una señal digital, ya que los valores que están preestablecidos, son finitos, no son ya todos posibles.

La señal ha quedado representada por un valor que durante la codificación (siguiente proceso de la conversión analógico digital) se transformará en una sucesión de ceros y unos, en una cadena de código binario.

Así pues, la señal digital que resulta tras la cuantificación es sensiblemente diferente a la señal eléctrica analógica que la originó, por lo que siempre va a existir una cierta diferencia entre ambas que es lo que se conoce como error de cuantificación, que se produce cuando el valor real de la muestra no equivale a ninguno de los escalones disponibles para su aproximación que es lo habitual. En ese caso, la distancia entre el valor real y el que se toma como aproximación determina un error. Este error de cuantificación se convierte en un ruido cuando se reproduzca la señal tras el proceso de decodificación digital.

Codificación: La codificación consiste en la traducción de los valores de tensión eléctrica analógicos que ya han sido cuantificados o ponderados al sistema binario, mediante códigos preestablecidos. La señal analógica va a quedar transformada en un tren o cadena de impulsos digitales (sucesión de ceros y unos).

Existen dos planteamientos de la digitalización de la señal de televisión en color:

- **Codificación de la señal compuesta:** Se codifica la señal analógica en función del estándar de televisión que haya en el país donde se está realizando la codificación: NTSC, PAL, SECAM. No permite la compatibilidad entre los estándares.
- **Codificación por componentes:** Se digitaliza la señal analógica utilizando su división por componentes:
    - Luminancia (Y) o brillo de la señal de vídeo por un lado.
    - Crominancia por el otro. Después de registrados los valores de los colores Red/Blue/Green o valores RGB, se definen las subportadoras de color restando los valores de intensidad de Red y de Blue de la luminancia o brillo de la señal (Y): R-Y y B-Y. La principal ventaja es que, por primera vez, se pueden respetar la compatibilidad entre estándares. Sólo se requiere un conversor D/A específico para cada estándar, y se mantiene así la compatibilidad con los estándares analógicos.

2.3.2 La compresión de vídeo digital

La compresión es el proceso de eliminar o reestructurar datos para reducir el tamaño de un archivo. Los archivos de vídeo digital son muy grandes y requieren velocidades de transferencia de datos muy altos para la captura y la proyección.

Con técnicas de compresión eficaces se pueden reducir considerablemente el tamaño del archivo sin que ello afecte la calidad de la imagen. Sin embargo, la calidad del video puede verse afectada si se reduce en exceso el tamaño, aumentando el nivel de compresión de la técnica que se utilice.

Existen diferentes técnicas de compresión, tanto patentadas como estándar. En la actualidad la mayoría de los proveedores de video en red utilizan técnicas de compresión estándar.

Los estándares son importantes para asegurar la compatibilidad y la interoperabilidad, tienen un papel especialmente relevante en la compresión de video, puesto que éste se puede utilizar para varias finalidades.

Algunas de las estrategias básicas para reducir la cantidad de datos de un vídeo son: comprimir los datos, reducir las dimensiones de imagen del vídeo o reducir la velocidad de fotogramas del vídeo capturado.

2.3.3 Conceptos básicos para la compresión

2.3.3.1 RGB

Este formato es el equivalente a un .wav en audio o un .bmp en fotografía, es decir, se captura con el 100% de calidad y sin compresión alguna. Las siglas RGB

provienen del inglés, Red (rojo), Green (verde) y Blue (azul) haciendo referencia a la característica del ojo humano de percibir tan sólo estos tres colores básicos y el resto de la gama de color se crea a partir de la combinación de estos tres.

Aunque con este formato se obtiene la máxima calidad, los altos requerimientos de espacio y de ancho de banda, para no perder cuadros (frames) en la captura harán que no se utilice en la mayoría de las ocasiones.

Cuando se hace referencia al formato RGB se suele especificar también la profundidad de color, que se expresa en bits. Así, al decir RGB24 se está indicando que el vídeo tiene una profundidad de color de 24 bits (16.777.216 colores) Los valores posibles son RGB32, RGB24, RGB16 y RGB15.

2.3.3.2 YUV

Otra característica del ojo humano, es que es más sensible a la luminosidad (cantidad de luz por pixel) que al color. Si se centra la compresión en el color preservando la luminosidad, se consigue un primer paso en la reducción del tamaño de la captura sin afectar demasiado a la calidad. La "Y" hace referencia a la luminosidad, mientras que "U" y "V" a la crominancia o color.

Este sistema de compresión suele venir por defecto en un buen número de tarjetas capturadoras sin necesidad de instalar ningún códec y la mayor parte de los códecs de compresión también lo utilizan como base para compresiones mayores.

Hay una gran variedad de formatos YUV, algunos presentan diferencias muy sutiles y otras más importantes. El principal aspecto a tener en cuenta cuando se comprime en YUV es el "subsampling".

Los formatos YUV más comunes son:

- **YUV2:** Es el más utilizado y equivale al 4:2:2. Se obtiene la luminosidad de cada píxel y el color de cada dos. Se obtiene una calidad muy cercana al RGB24 con una compresión bastante buena.
- **YUV8:** Se elimina por completo la información de color dejando tan sólo la luminosidad. Es la famosa escala de grises de 8 bits por píxel y 256 tonos.
- **YUV9:** La luminosidad se toma de cada píxel, mientras que para el color se obtiene un valor medio de un matriz de 3x3 pixels
- **BTYUV:** Es equivalente a 4:1:1. La luminosidad de cada píxel se conserva. Para el color, se agrupan 4 píxeles por cada línea y se obtiene la media.
- **YUV12:** Este formato es muy empleado en compresión MPEG y explica el porqué de la frecuente pixelación con flujos de datos bajos, ya que se obtiene un valor medio de luminosidad y color por cada matriz de 2x2 píxeles.

  Hay muchas más variedades y en numerosas ocasiones un mismo tipo de YUV recibe nombres distintos. De hecho, la compresión YUV puede llamarse YCC, YCbCr, YPbPr o YIQ atendiendo a diversas consideraciones en el tratamiento del color.

El "Subsampling" (sub-muestreo) consiste en reducir la información de color preservando intacta la luminosidad. Eso se expresa de la siguiente manera:

- 4:4:4 mantiene intacta tanto la información de la luminosidad (primer "4") como la del color (los otros dos "4"s).
- 4:2:2 reduce el muestreo del color a la mitad.
- 4:1:1 reduce el muestreo de color a la cuarta parte.

- 4:2:0 elimina uno de los valores de color dejando el otro valor en la mitad.

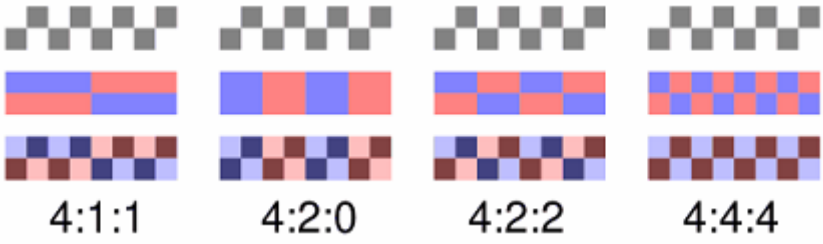


Figura II. 5 Ejemplo de los distintos niveles de subsampling

Los valores empleados en la compresión JPG y MPEG son 4:1:1 y 4:2:0 respectivamente. Para saber el por qué se le llama "sub-muestreo" hay que recordar que en un televisor la imagen se forma de manera entrelazada entre 625 líneas horizontales (525 para NTSC). Por tanto, el "submuestreo" hace referencia a que para comprimir la imagen, se utilizan tan solo el muestreo vertical, puesto que las líneas son horizontales y se cuentan de "arriba a abajo".

Figura II. 6 Imagen sin subsampling

Figura II. 7 Imagen con subsampling aplicado

2.3.4 Tipos de compresión

Existen dos tipos de compresión de vídeo intra-frame e inter-frame:

Compresión Intra-Frame: O compresión espacial, comprime cada fotograma por separado. El Intra-Fame son las imágenes completas. Este método es de mayor

calidad, lo que hace que ocupe más espacio, un ejemplo de tipo de compresión de este método es AVI.

Codificación Inter-Frame: O compresión temporal**,** aprovecha la ventaja que existe cuando las imágenes sucesivas son similares. En lugar de enviar la información de cada imagen por separado, el codificador inter envía la diferencia existente entre la imagen previa y la actual en forma de codificación diferencial. De esta forma, se elimina la redundancia temporal, usando información de las imágenes ya enviadas y enviando únicamente las zonas de la imagen que han cambiado de un fotograma a otro. Este tipo de compresión necesita de una imagen, la cual fue almacenada con anterioridad (key frame) para luego ser comparada con imágenes sucesivas y de forma similar, se requiere de una imagen previamente almacenada para que el decodificador desarrolle las imágenes siguientes.

MPEG es el principal ejemplo de este tipo de compresores y se caracteriza porque puede mantener una alta calidad en la imagen a pesar de utilizar ratios de compresión mayores que la compresión intra.

Los datos que se generan al hacer la diferencia entre dos imágenes también se pueden tratar como una nueva imagen (la cual se puede someter al mismo tratamiento de transformación utilizado en la compresión espacial).

El compresor retiene toda la información de la secuencia cada cierta cantidad de fotogramas y servirá de referencia a la hora de saber qué tipo de información debe desechar el compresor en las secuencias posteriores hasta encontrar un nuevo fotograma de referencia.

A las imágenes obtenidas como diferencia de la imagen no comprimida y la imagen de referencia se las conoce como delta frames.

Una secuencia de imágenes que está constituida por una imagen inicial (key frame) “I” y las siguientes imágenes “P” (delta frames) hasta el comienzo de otra imagen “I”, se denomina GOP (Group Of Pictures) o grupo de imágenes. Para factores de compresión altos se utiliza un número grande de delta frames, haciendo que las GOPs aumenten de tamaño considerablemente; sin embargo un GOP grande evita recuperar eficazmente una transmisión que ha llegado con errores.

En el caso de objetos en movimiento, puede que su apariencia no cambie mucho entre imágenes, pero la representación de los bordes si cambia considerablemente.

2.3.5 Formatos de audio y video

Un formato de video digital, es la manera en que se guardan los datos en el archivo, esta forma puede cumplir diferentes requisitos según el uso para el que este diseñado.

AVI: Audio Video Interleave, en español Audio y Vídeo Entrelazado, es el formato de Windows desarrollado por Microsoft. Las pistas de audio y vídeo se encuentran grabadas de forma consecutiva en varias capas. Se ha ido alternando la grabación entre imagen y sonido, pero de una forma tan rápida que nuestros sentidos, tanto el oído cómo vista lo perciben de forma paralela. AVI es un formato de archivo que puede guardar datos en su interior codificados de diversas maneras y que utiliza diferentes códecs que aplican diferentes factores de compresión.

MPEG: Motion Picture Experts Group, en español Grupo de Expertos de Imágenes en Movimiento, es un grupo de estándares empleados para la codificación de información audiovisual, incluyendo video y audio en formato digital comprimido. El sistema MPEG surgió como una respuesta a las necesidades de transmitir señales de televisión y vídeo por redes digitales y está pensado para funcionar con aparatos diseñados específicamente para esta misión que permitan comprimir y descomprimir las imágenes a gran velocidad.

Este formato se clasifica en: MPEG-1, MEPG-2, MEPG-3 Y MPEG-4

- MPEG-1: Este formato de compresión de vídeo digital, surgió durante el año 1991. Su calidad se parece al del sistema VHS. La principal finalidad de este tipo de formato de compresión fue el de poder colocar el vídeo digital en un soporte muy conocido, el CD-ROM. Su tamaño es de 1,5 mega bits por segundo y se presentaba a una resolución de 352 x 240 píxeles NTSC o 352 x 288 en PAL.
- MPEG-2: Aparece en 1994 y es uno de los formatos de compresión que ofrece mayor calidad de imagen, alcanza a su vez una velocidad en la transmisión de datos de 3 a 10 Mbits de ancho de banda. Este formato ofrece la transmisión de ficheros de vídeo digital a pantalla completa o broadcast. El formato MPEG2, es el utilizado por la televisión digital y para la codificación del DVD de vídeo. Trabaja con resoluciones desde 352 x 480 y 1920 x 1080 píxeles o 720x576 (PAL) y 720x480 (NTSC).
- MPEG-3: Se desarrolló para la televisión digital de alta calidad aunque el formato MPEG-2 también cumplía perfectamente esta función. El formato

MPEG3 tiene mayor ancho de banda que el MPEG-2 y se optó por la utilización finalmente del formato MPEG-2. Por este motivo el proyecto orientado en el MPEG-3 se abandonó.

- MPEG-4: Uno de los códecs utilizados en este formato son los DivX y XviD. Una de las grandes ventajas que ofrece este formato es una muy buena calidad, muy parecida al del formato DVD, a cambio de un factor de compresión mucho más elevado que otros formatos, dando como resultado archivos o ficheros más comprimidos que otros e ideales para poder transmitir los datos a través de Internet. Utiliza una resolución de 176 x 144 píxeles.
- MPEG-7: Este formato está en proyecto, cómo el estándar que más se utilizará en internet y televisiones interactivas. Este formato codificará además de la imagen y sonido datos en lenguaje XML. MPEG-7 servirá de gran ayuda para el avance de la nueva televisión interactiva con introducción de buscadores de contenidos, búsquedas de audiovisuales, etc.

MOV: Un archivo QuickTime MOV contiene una o más pistas, las cuales pueden ser audio, video, efectos o subtítulos.

La principal desventaja es que este formato es desarrollado por Apple y requiere la instalación de QuickTime para poder reproducirlo bajo sistemas operativos Windows.

ASF: Advanced Systems Format es el formato propiedad de Microsoft para contener audio y video.

Este formato no especifica cómo debe codificarse el audio o el video, en su lugar sólo especifica la estructura de la pista de audio/video. Esto quiere decir que los archivos ASF pueden contener pistas comprimidas con cualquier códec y aun así seguirían siendo archivos en formato ASF. Esto es algo similar a lo que se hace con los formatos QuickTime, AVI, y Ogg.

Los formatos más comunes contenidos por ASF son Windows Media Audio (.wma) y Windows Media Video (.wmv).

Windows Media Video: El formato .wmv es una extensión que no tiene diferencia con los archivos .asf. Estos usan el formato de fichero estándar de Windows Media. Los ficheros con extensión .asf normalmente son utilizados en contenidos basados en Windows Media usando las Herramientas Windows Media 4.0. Los ficheros .wma y .wmv se introdujeron como un convenio con la versión 7 para posibilitar a los usuarios la fácil diferenciación entre los ficheros de audio .wma y los de video .wmv. Sin embargo, no hay ninguna limitación en el formato, y las extensiones pueden usarse intercambiablemente. Algunas herramientas y servicios que fueron creados para ser usados con las versiones anteriores de las Tecnologías Windows Media requieren la extensión de .asf en orden para aceptar el contenido. Se puede simplemente renombrar cualquier archivo .wma o .wmv para usar la extensión de .asf y usarlos con esas herramientas.

WMA: Es un formato de compresión con pérdidas, utiliza un sistema de compresión muy parecido al MPEG. WMA, también elimina el sonido sobrante y algunas frecuencias que puedan resultar muy bajas o altas para nuestros oídos, pero con un

factor de compresión de alta calidad. En algunos aspectos WMA, comprime mejor que el códec MP3. Este formato está desarrollado por Microsoft.

MP3: Permite hacer una pista de audio en formato Compact Disk, ocupando muy poco espacio y sin apenas pérdida de calidad. El formato de compresión MP3, se utiliza en los reproductores de audio, CD portátiles, DVD doméstico, pen-drives, etc.

AC3: Es el sonido Dolby Digital, puede ser un sonido estéreo o sonido envolvente. Este sonido presenta seis canales de 16 bits y 48 KHZ, más frecuencia que un compact Disk. Un compact Disk tiene 44,1 KHZ. Esto da lugar a que una pista de audio ocupará 4 veces más que una de audio con MP3.

2.3.6 Códecs

La palabra códec se traduce de las palabras codificador y decodificador. Un códec es una serie de funciones algorítmicas necesarias para comprimir y descomprimir los datos de audio y vídeo de un archivo. Significa que si al reproducir un vídeo digital no se tiene instalado el códec con el que se han codificado los datos no se podrá visualizar correctamente en el computador.

Existen gran variedad de códecs entre los más importantes podemos destacar los siguientes:

Ligos Indeo: Códec de vídeo desarrollado inicialmente por Intel, pero adquirido posteriormente por Ligos. Fue uno de los primeros códecs que permitieron reproducir vídeo a plena velocidad sin requerir aceleración extra de hardware.

Muy usado para distribuir vídeo por Internet. Este códec emplea un sistema progresivo de descarga que se adapta al ancho de banda y flujos de señal. Su tecnología está basada en un compresor/descompresor en forma de controlador de software que comprime los datos de vídeo digital para almacenarlos y los descomprime para su reproducción en un PC multimedia.

No comprime los datos de audio y permite bajo bitrate y baja resolución.

Cinepak: Su fabricante es Radius Corp. aunque es un codec algo antiguo todavía proporciona video de buena calidad a pantalla completa alta velocidad de muestreo y profundidad de muestra. Los datos de audio no se comprimen y consigue en resumen mejor calidad de color, así como un bajo bitrate, lo que le hace que tenga buenas características de reproducción.

Es un compresor altamente compatible, porque además de venir en el paquete de códecs de Windows, es compatible con el Quicktime de Apple. Trabaja comprimiendo temporalmente con un ratio ajustable, que proporciona una calidad muy alta, pero con archivos de un tamaño bastante elevado. Se utiliza con ratios de compresión alta para lograr archivos de calidad media para presentaciones multimedia o de CD interactivos.

La compresión es algo lenta pero es totalmente asimétrica respecto a la decodificación. Al reproducir desde CD-ROM, se obtienen muy buenos resultados trabajando a 320x240 píxeles de tamaño de cuadro y con una velocidad de reproducción de 15 a 25 imágenes por segundo. Este codec adapta la calidad en función de los ajustes que se hayan realizado para la compresión.

DivX: Es uno de los códecs más utilizados. Una de las grandes ventajas que presenta es su forma de compresión, manteniendo un alto grado de calidad en la imagen y un bajo peso en el archivo.

Este formato es muy utilizado para páginas Web en Internet. Por ejemplo con el método de compresión que nos presenta DivX, podemos disponer de una película completa de 2 horas de duración en un CD convencional de 700MB. Este códec fue creado por un grupo de hackers y se basa en el formato estándar de Microsoft el MPEG-4 y de uso gratuito. Más adelante surgió una nueva versión DivX 5, con funciones más avanzadas de compresión, pasando a ser una versión profesional no gratuita.

Xvid: Códec que a diferencia del DivX, este es "Open Source" de código abierto. En el colaboran y trabajan muchos programadores sin ninguna finalidad económica. Este códec funciona con un algoritmo de compresión para mejorar la calidad del vídeo y se presenta frente al DivX, cómo un posible sucesor o como el primer competidor. El codec Xvid, apareció durante el período 2004 y rápidamente fue muy bien acogido por los usuarios de la red, debido a la gran calidad que ofrece. Puede llegar a codificar a través de diferentes técnicas con la finalidad de mejorar la imagen. Xvid respecto al DviX, comprime de una forma en la cual ofrece bordes y figuras muy nítidas, este codec se está introduciendo de una forma muy fuerte para compartir vídeo a través de internet.

CAPÍTULO III

TARJETAS SINTONIZADORAS O CAPTURADORAS DE TELEVISIÓN

3.1. INTRODUCCIÓN

Una tarjeta sintonizadora o capturadora es un periférico que permite ver los distintos tipos de televisión o señales de video en la pantalla del computador, mediante programas instalados que ayudan al usuario a utilizar la tarjeta. La visualización se puede efectuar a pantalla completa o en modo ventana.

Este periférico puede ser una tarjeta de expansión, generalmente de tipo PCI, o bien un dispositivo externo que se conecta al puerto USB. Los modelos externos codifican la grabación por software; es decir, que es el procesador del ordenador quien realmente hace todo el trabajo. En cambio algunos modelos PCI realizan la

codificación de la grabación por hardware; es decir que es la propia tarjeta la hace, liberando de esa tarea al procesador del computador para dar mayor rendimiento a la máquina.

Una capturadora puede tener diferentes tipos de entradas y salidas, tanto de Video como de Audio. Estas pueden ser:

- Entradas de Video: S-Video (De 4 o de 7 pines), Video RCA, entrada de TV (UHF/VHF) y HDMI.
- Salidas de Video: S-Video (De 4 o de 7 pines), Video RCA, HDMI y VGA (Video Graphics Array) o DVI (Digital Visual Interface)
- Entrada y Salida de Audio: Miniplug estéreo.
- Además entrada de: FM Radio, por medio de un RCA con una frecuencia de 87.50MHz a 108.00 MHz.

Por medio de todas estas entradas y salidas, puede utilizarse una sintonizadora o capturadora para múltiples propósitos. El usuario, utilizando estas entradas y salidas puede transformar su computadora en un centro multimedia.

Las aplicaciones más comunes para las capturadoras o sintonizadoraspueden ser:

- Sintonización de Televisión. El usuario puede transformar su Pc en un TV, por ejemplo: almacenar en el disco duro del PC un programa de televisión favorito.
- Digitalización de VHS. Por medio de la entrada RCA o de S-Video, que son las salidas que comúnmente suelen tener las Video Caseteras reproductoras de los VHS, la capturadora es capaz de grabar esa señal de video y digitalizarla,

exportándola a un formato AVI o MPEG. De esta manera con el video digitalizado el usuario puede grabarlo en un DVD.

- Utilizar el monitor del PC como TV. Con las entradas que posee una capturadora se puede conectar una consola de video juegos, como por ejemplo un Play Station. También puede conectarse un video filmador que posea salida de audio y video.
- Utilizar la señal de video proporcionada por una Cámara Digital como la Cámara Web. Al conectar una Cámara que posea salida de video, en la entrada de la capturadora esta será identificada por el programa a utilizar, por ejemplo, Windows Live Mesenger que identificará la capturadora en su configuración.
- Utilizar la señal de video capturada y transmitirla en vivo por internet. A modo de ejemplo se puede dar el sitio web “www.rojadirecta.org” que transmite todos los partidos de futbol en vivo.
- Grabar audio de la Radio FM. Todas aquellas placas sintonizadoras que posean radio FM, por medio de software, permiten grabar el audio al disco rígido del PC en formatos de audio: Mp3, WAV, etc. A modo de ejemplo se podrá grabar un programa de radio favorita para tenerlo almacenado.

3.2. CARACTERÍSTICAS DE UNA SINTONIZADORA O CAPTURADORA

Una capturadora se puede diferenciar de otra por las cantidades y tipos de entradas que posee, principalmente por la resolución que ofrece.

En los últimos años se han popularizado las tarjetas sintonizadoras de TV, mediante las cuales se pueden ver la TV en el monitor del PC a pantalla completa, o en una

ventana de Windows, mientras se realiza otras tareas en el computador. Algunas disponen de sintonizadora de radio FM, y hasta mando a distancia.

Pero no son las únicas cualidades y funciones de estas tarjetas. Por lo general, dichas tarjetas también son capturadoras de video, es decir, se puede grabar en un fichero AVI, en el disco duro una secuencia de video, con su sonido correspondiente.

Existen otras tarjetas que solo son capturadoras, es decir, no sintonizan, por lo que capturan la señal de video proveniente de una de sus entradas. Estas tarjetas son las usadas por las cámaras de video para Chat.

3.3. TIPOS DE TARJETAS SINTONIZADORAS O CAPTURADORAS

Actualmente existen distintos tipos de tarjetas sintonizadoras, según el tipo de emisión de televisión que se requiera recibir en el computador:

- **Analógicas:** Sintonizan los canales analógicos recibidos por antena (TV Publica) y/o por cable (Multicanal).
- **Digitales:** Las de tipo DVB-T sintonizan los canales de la televisión digital terrestre TDT, que se recibe por antena. Las de tipo DVB-C sintonizan los canales de la televisión digital por cable, pero no los de TDT. Actualmente no hay modelos "combinados" DVB-T/C.
- **Satélite:** Sintonizan los canales de la televisión recibidos por antena parabólica (por ejemplo, del satélite DirectTV).

También existen modelos híbridos, que son capaces de sintonizar al mismo tiempo dos o más de estos tipos de emisión. Algunos modelos añaden también la sintonización de radio FM.

Las sintonizadoras analógicas soportan un sistema de color determinado: PAL, SECAM o NTSC.

3.4. SOFTWARE DE CAPTURA DE VIDEO Y COMPRESIÓN

Las tarjetas sintonizadoras se venden junto a sus drivers y un software que permite la sintonización, memorizado, visualización y grabación. También existe software gratuito de terceros tanto para Windows como para Linux. Estos programas funcionan con la mayoría de tarjetas sintonizadoras y en muchos casos, mejora la calidad de la visualización y de la grabación obtenida por el software original de la sintonizadora, ejemplos de estos softwares son:

- **Dscaler:** Es un programa para aprovechar las prestaciones de una sintonizadora de TV y es compatible con una gran variedad de tarjetas. Dscaler posee una interfaz intuitiva se destaca por la calidad de imagen que logra y también por la posibilidad de grabar video.
- **Kastor! TV:** También conocido como K!TV, es una aplicación desarrollada bajo la licencia GNU y compatible con un gran número de tarjetas de televisión. Incorpora una interfaz multilingue y soporta varios estándares de audio y vídeo.
- **MythTV:** Es una colección de herramientas software para la televisión bajo Linux. Permite a los usuarios y desarrolladores incorporar nuevas e innovadores maneras de extender sus funcionalidades.
- **TvTime:** Es un reproductor de televisión de alta calidad, recibe la señal de una tarjeta capturadora de televisión, la procesa y la muestra en la pantalla.

Soporta multitud de algoritmos y filtros de video consiguiendo 59.94 frames por segundo para NTSC, y 50 frames por segundo para PAL.

3.5. TARJETAS SINTONIZADORAS MÁS COMUNES EN EL MERCADO

En el mercado se encuentra una gran gama de modelos y fabricantes de sintonizadoras de TV, entre los más conocidos podemos enumerar los siguientes:

Tarjeta Kworld USB Analogica UB405-A

Figura III. 8 Tarjeta Kworld UB405-A

Características pricipales:

- Soporte de los sistemas analógicos televisivos de todo el mundo.
- Software TiVme exclusivo de Kworl.
- Permite ver y grabar programas de televisión analógicas.
- Soporte de grabación en los formatos MPEG 1, VCD, MPEG 2, DVD, MPEG 4, MPEG 4_PSP, MPEG4_iPod, H.264_PSP y H.264_iPod
- Permite decodificar el audio/video analógico de todo el mundo.

- Permite usar diferentes modos como SAP, estéreo y Mono.
- Compatible con Windows Xp, Vista y Windows 7.
- Brinda movilidad y portabilidad al ser una tarjeta tipo USB.

Tarjeta DUB-T210 TV Tuner/Mpeg Encoder

Figura III. 9 Tarjeta DUB-T210

Características principales:

- Sintonizador de TV.
- Control remoto incluido.
- DUB-T210 incluye un puerto USB 2.0 de alta velocidad.
- Soporte grabación en formato digital AVI o bien comprimidos en formatos MPEG en el disco duro o CD ROMs.
- Permite al usuario capturar 30 cuadros por segundo en una resolución de imagen de alta calidad (720 x 480 NTSC).

- Se puede utilizar bajo el sistema operativo Windows XP, 2000, no soporta Windows Vista.

Tarjeta Hauppauge Win TV PVR - 150

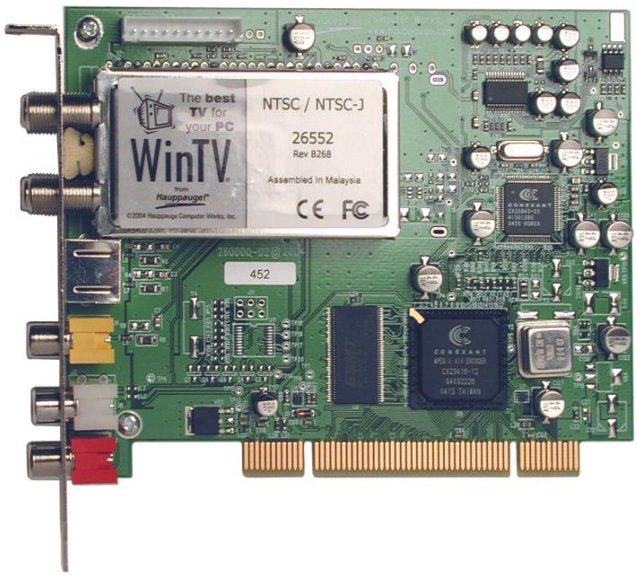


Figura III. 10 Tarjeta Hauppauge Win TV PVR - 150

Características principales:

- Tarjeta de vídeo PCI.
- Cuenta con hardware de vídeo acelerador y compresión de audio.
- Muy estable, de fácil manejo.
- Posee un sistema incorporado para codificación por hardware en MPEG-2.
- Compatible con Windows Xp, Vista y Windows 7, Linux en todas sus variantes.
- Sistema de TV NTSC, PAL y SECAM
- Entradas para TV/FM, Video/Audio, S-Video/Compuesto

Tarjeta Zogis Real Angel 220 Sop

Figura III. 11 Tarjeta Zogis Real Angel 220

Características principales:

- Soporte de los Sistemas de TV NTSC, PAL y SECAM.
- Posee entradas de TV/FM, Video/Audio, S-Video/Compuesto.
- Soporta recepción de FM.
- Apagado del PC a través de control remoto.
- Posee un sistema incorporado para codificación por hardware en MPEG-2.
- Compatible con Windows XP SP2 / Vista.

CAPÍTULO IV

DIRECTX

4.1 INTRODUCCIÓN

La llegada de las capacidades gráficas a los ordenadores personales, permitió la posibilidad de ver imágenes en movimiento. Los primeros sistemas que cumplían este propósito eran normalmente lentos, de baja calidad y además las imágenes almacenadas ocupaban muchísimo espacio. El primer sistema fue el formato FLI de Autodesk. Este formato en sus orígenes era una secuencia de imágenes GIF a 320x200 y 256 colores sin sonido. El sistema de compresión usado era muy simple: tomaba la primera imagen (fotograma) almacenando solo las variaciones que existían de la siguiente imagen. Con esto se conseguía un ratio de compresión de

1:2 aproximadamente. Este formato posteriormente evolucionó admitiendo distintas resoluciones y más profundidad de color, manteniendo el mismo sistema de compresión sin pérdida.

Compresión, es el proceso de eliminación o reestructuración de los datos para disminuir el tamaño de un archivo. Los archivos de video digital son muy grandes, requiriendo gran velocidad de transferencia de datos en la lectura y reproducción.

Video For Windows fue un framework multimedia que añadió por primera vez a Microsoft Windows el soporte para operar con ficheros de video.

Video For Windows fue lanzado en 1992. En su versión gratuita, era en principio un programa instalable para Windows 3.1 y 3.11 que incluía compatibilidad con ficheros AVI y un conjunto de herramientas para reproducir video, entre otras una actualización de Media Player, y un conjunto de códecs de video, con RLE y Video1 entre otros. Versiones posteriores añadirían Cinepak e Indeo.

Más adelante, con la llegada de Windows 95, Video for Windows se incluyó como parte del sistema operativo, no siendo necesaria ya su instalación por separado.

En 1996, se implantó ActiveMovie para sustituir a Video for Windows en gran parte de las tareas de video, salvo la de captura, que seguía requiriendo el uso de Video for Windows. Finalmente, ActiveMovie y Video for Windows serían sustituidos en 1997 por DirectShow 1.0.

4.2 CONCEPTO

DirectX® de Microsoft® es el software que permite manipular el hardware del sistema a bajo nivel.

DirectX proporciona acceso de bajo nivel al hardware multimedia de forma independiente del dispositivo. Adicionalmente, aprovecha los últimos desarrollos en hardware a medida que se van originando como es el caso de la tecnología MMX de Intel®. Esto se logra simplemente instalando la última versión de DirectX.

MMX es un Conjunto de instrucciones SIMD diseñado por Intel e introducido en 1997 en sus microprocesadores Pentium MMX. Fue desarrollado a partir de un set introducido en el Intel i860. Ha sido soportado por la mayoría de fabricantes de micros x86 desde entonces.

DirectX incluye varios componentes (Figura IV. 12) entre los que se encuentran:

- **DirectDraw®:** Este proporciona animación realista usando intercambio de páginas de video, acceso a coprocesadores gráficos especializados y administración de la memoria de video. También sirve de base para otros componentes como DirectShow® y Direct3D®.
- **Direct3D®:** Proporciona interfaces de alto y bajo nivel para generar polígonos con texturas en 3D por software y por hardware.
- **DirectSound®:** Proporciona sonido estéreo y 3D con mezcla de sonido por hardware, así como administración de la memoria de la tarjeta de sonido.

- **DirectPlay®:** Incluye servicios transparentes de mensajería independientes del medio para crear juegos con varios jugadores, así como las funciones necesarias para organizar y ejecutar un juego multijugador.
- **DirectShow®:** Proporciona una interfaz para el manejo de flujos multimedia provenientes de archivos o dispositivos de adquisición de audio y video.

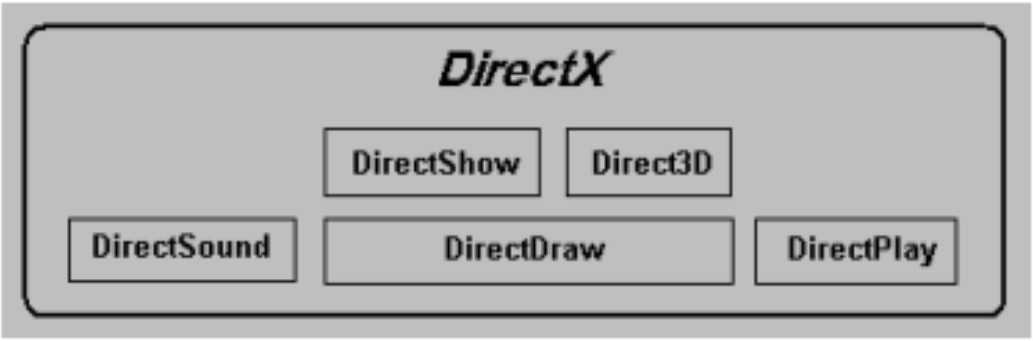


Figura IV. 12 Componentes de DirectX

Directshow es la herramienta a utilizar en el presente trabajo, y para su comprensión es necesario revisar los conceptos sobre el modelo de objetos componentes COM desarrollado por Microsoft.

4.3 EL MODELO DE COMPONENTES COM

COM (Component Object Model) es un estándar creado por Microsoft que define a nivel binario como los objetos se deben crear, destruir e interactuar entre ellos. El hecho de que sea un estándar a nivel binario implica que es independiente del lenguaje que se use para trabajar con los objetos, siempre y cuando el lenguaje en cuestión sea capaz de manejar apuntadores a funciones.

Los componentes de software COM son independientes de cualquier aplicación y residirán en el sistema en forma de DLL (Dynamic Link Library). Cada componente

dispone de un identificador global único o GUID (Globally Unique Identifier) que lo caracteriza (este identificador consiste de un entero de 128 bits y se puede generar mediante el programa guidgen.exe). Si el componente está registrado en el sistema, entonces se podrá crear un objeto concreto de esa clase. Estar registrado en el sistema significa que el sistema operativo conoce donde reside "físicamente" la implementación de la clase identificada por el GUID. En el sistema Windows esta información se almacena en una base de datos centralizada llamada registro. Este registro se utiliza para configurar el software y hardware que se ejecute en el ambiente Windows. De esta manera, se consigue disponer de toda una librería de componentes que se podrá usar libremente, siempre y cuando estén registrados en el sistema.

Una interface es un conjunto de métodos (funciones) y atributos (datos) que tienen una fuerte relación lógica entre ellos. Un componente es la implementación de una o más interfaces y queda definida por las interfaces que implementa. La herencia de un componente se concibe a nivel de interfaces. Así todas las interfaces derivan de la interface IUnknown. Esta interface se encarga de la gestión de memoria de los objetos. La gestión de memoria se basa en un contador de referencias. La interface IUnknown está formada por tres métodos AddRef, Release y QueryInterface para controlar este contador. Las dos primeras permiten llevar la cuenta de si el componente está siendo utilizado y cuantos clientes lo están utilizando; de esta forma el sistema puede descargar un componente cuando no se esté utilizando. La tercera permite averiguar si el objeto en cuestión implementa una interface en particular o no, además de obtener un acceso al objeto a través de esa interface.

4.3.1 Creación y gestión de objetos

Los objetos se crean mediante la función CoCreateInstance, y se acceden a través de apuntadores a interfaces. Inicialmente, se obtiene un apuntador a una interface a raíz de la creación del propio objeto. Posteriormente, se puede acceder a las otras interfaces que se implementen con el método QueryInterface de ésta. Por ejemplo, si se tiene un componente identificado por el GUID GUID_MYCOMPONENT, que implementa las interfaces InterfaceA e InterfaceB se puede escribir el siguiente código:

```
InterfaceA *pIA;
InterfaceB *pIB;

CoInitialize(NULL);
CoCreateInstance(GUID_MYCOMPONENT, NULL, CLSCTX_INPROC, GUID_INTERFACEA,
(LPVOID *) &pIA);

// Ahora se puede acceder a los métodos y atributos de la InterfaceA a través de pIA
pIA->QueryInterface(GUID_INTERFACEB, (LPVOID *) &pIB);

// ... Ahora también se puede acceder a la InterfaceB a través de pIB
pIA->Release();

// ... Ahora el objeto aún no se ha destruido porque aún se tiene una interfase activa
pIB->Release();

// ... Ahora sí que se destruye el objeto
CoUninitialize();
```

Entonces se puede decir que:

- Primero, se tiene que comunicar al sistema que la aplicación va a ser cliente (o usuario) de objetos COM mediante la función CoInitialize, que se deberá

llamar al principio del programa. De la misma manera, hay que comunicar que se deja de ser clientes mediante CoUninitialize.

- Por otro lado, se observa que las interfaces también tienen un GUID asociado que las identifica de forma universalmente única, es decir, no debería existir ninguna otra interface que tenga el mismo GUID.
- El parámetro CLSCTX_INPROC especifica el contexto de ejecución del objeto, es decir, si se ejecutará dentro del mismo espacio de proceso de la aplicación, fuera de éste, etc. En general siempre se especificara que se ejecute dentro del proceso planteado.
- Las referencias a CoCreateInstance y QueryInterface ya incrementan el contador de referencias, por lo que no es necesario hacerlo explícitamente con AddRef. En cambio, sí es necesario liberar los apuntadores a interface llamando a Release.

4.3.2 Gestión de errores

En el modelo COM se ha estandarizado la gestión de errores. Todas las funciones COM y métodos de objetos COM devuelven un valor de tipo HRESULT con un código de error. Si todo ha funcionado correctamente, este valor es S_OK. En cambio, si ha habido algún error, éste código dependerá de la función o método que sea invocado, ya que cada una puede generar tipos diferentes de errores. Se debe consultar la ayuda si se desea hacer un tratamiento exhaustivo de los diferentes tipos de errores.

En general, habrá que controlar como mínimo el resultado de la función CoCreateInstance y del método IUnknown::QueryInterface, ya que éstos devuelven

apuntadores a interfaces COM a través de las que accederemos al objeto. Si hay algún error en estos casos, el apuntador a interfase que se obtendra será erróneo, y en el momento en que se quiera llamar a alguno de sus métodos provocará un acceso a memoria no válido.

Explicado el funcionamiento de COM, se puede ver las características de DirectShow.

4.4 DIRECTSHOW

DirectShow es un paquete para desarrollar software que permite obtener flujos de datos de video y/o audio a partir de archivos o desde dispositivos de hardware como tarjetas de sonido y de video conectados a la computadora.

Este soporta una amplia variedad de formatos, incluyendo ASF (Advanced Streaming Format), MPEG (Motion Picture Experts Group), AVI (Audio-Video Interleaved), MP3 (MPEG Audio Layer-3), y archivos WAV.

También soporta captura de video utilizando dispositivos WDM (Windows Driver Model) o antiguos dispositivos Video para Windows.

DirectShow está integrado junto con otras tecnologías DirectX. Detecta automáticamente y utiliza el hardware para la aceleración de audio y video cuando se encuentren disponibles en el sistema.

Adicionalmente permite la elaboración de componentes de software que realicen algún procesamiento (compresión, escalamiento, filtrado), sobre dicho flujo en tiempo real.

Las aplicaciones se basan en un sistema modular de componentes llamados filtros. El filtro es un objeto COM y se ubica en un arreglo denominado gráfico de filtros (filter graph) el cual proporciona además una idea global del funcionamiento e interacción entre filtros.

4.4.1 Gráfico de Filtros

El bloque de construcción básico de DirectShow es un componente de software llamado filtro. Un filtro generalmente realiza una sola operación en un flujo multimedia. Por ejemplo, hay filtros que realizan las siguientes tareas:

- Obtiene video desde un dispositivo de captura de video.
- Lee archivos.
- Decodifica un formato particular del flujo, como video MPEG-1.
- Pasa datos hacia la tarjeta de video o de sonido.

Un filtro recibe entrada (input) y producen salida (output). Por ejemplo, si un filtro decodifica video MPEG-1, la entrada es un flujo MPEG-codificado y la salida es un flujo de video RGB no comprimido. Para realizar una tarea dada, una aplicación conecta varios filtros donde la salida de un filtro es la entrada de otro.

Un conjunto de filtros conectados se denomina gráfico de filtros (filter graph). Como una ilustración de este concepto, la Figura IV. 13 muestra un gráfico de filtros para ejecutar un archivo AVI.

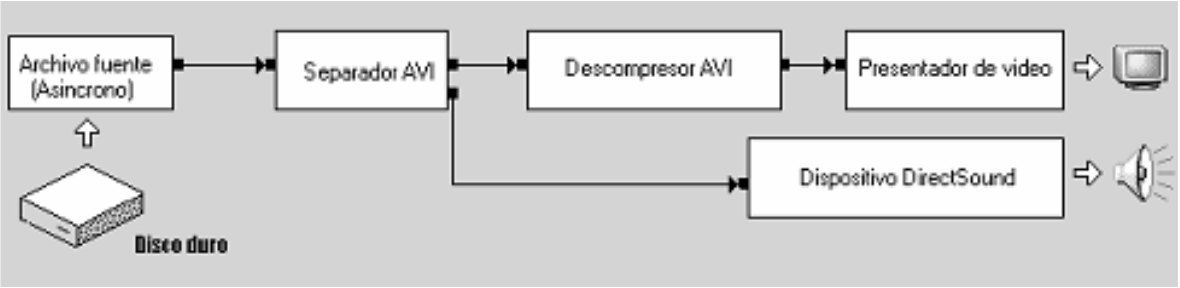


Figura IV. 13 Gráfico de filtros

La aplicación no tiene que manejar los filtros individuales en el gráfico de filtros. En lugar de eso, DirectShow provee de un componente llamado Manejador del Gráfico de Filtros (Filter Graph Manager).

El Manejador del Gráfico de filtros controla el flujo de datos en el gráfico. Una aplicación de alto nivel solo realiza llamadas tales como “Run” (para mover los datos en el gráfico) o “Stop” (para detener el flujo de datos). Si se requiere un control más directo de las operaciones sobre el flujo, se puede acceder directamente a los filtros a través de las interfaces COM.

El Manejador del gráfico de filtros pasa notificaciones de eventos a la aplicación, de modo que la aplicación puede responder a eventos tales como el fin de flujo.

4.4.2 Aplicaciones DirectShow

Una aplicación típica DirectShow realiza tres pasos básicos, como ilustra la Figura IV. 14

Crea una instancia del Manejador del Gráfico de filtros, usando la función CoCreateInstance.

1. Utiliza el Manejador del gráfico de filtros para construir un gráfico de filtros.

2. Controla el gráfico de filtros y responde a eventos.

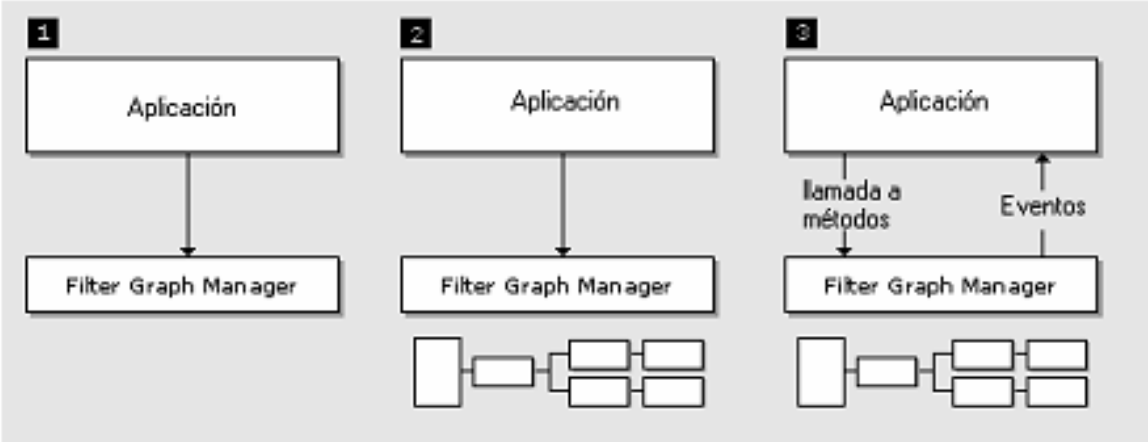


Figura IV. 14 Desarrollo de una aplicación en DirectShow

El Manejador del gráfico de filtros inspecciona la conexión de los filtros en el gráfico de filtros y controla el flujo de datos. Las aplicaciones del usuario controlan las actividades del gráfico de filtros mediante la comunicación con el Manejador del Gráfico de Filtros. Para elaborar un filtro se requiere el uso de programación COM y utilizar las clases base en las librerías de DirectShow implementadas en C#.

4.5 COMPONENTES DE DIRECTSHOW

Trabajar con multimedia presenta varios problemas principales:

- Los flujos multimedia contienen grandes cantidades de datos que deben ser procesados a altas velocidades.
- El audio, video y cualquier flujo adicional deben estar todos sincronizados para iniciar y detenerse todos al mismo tiempo y ejecutarse a la misma rapidez.

- Los flujos pueden venir desde muchas fuentes, incluyendo archivos multimedia locales, tarjetas de adquisición de video/audio, videocámaras y otros dispositivos.
- Los flujos vienen en una variedad de formatos, tales como AVI, ASF, MPEG, DV y MJPEG.

Para llevar a cabo la tarea de proveer flujos de video y audio de manera eficiente hacia las tarjetas gráficas y de sonido, DirectShow utiliza DirectDraw® y DirectSound®. La sincronización se lleva a cabo encapsulando los datos multimedia en muestras con información temporal.

Para manejar la variedad de fuentes, formatos y dispositivos de hardware, DirectShow utiliza una arquitectura modular en la cual componentes llamados filtros, pueden ser mezclados y acoplados para realizar tareas distintas.

DirectShow incluye filtros que soportan captura de multimedia y dispositivos sintonizadores basados en Windows Driver Model (WDM) así como también filtros que soportan Video for Windows (VfW). La Figura IV. 15 muestra la relación entre una aplicación, los componentes DirectShow, y algunos de los componentes hardware y software que DirectShow soporta.

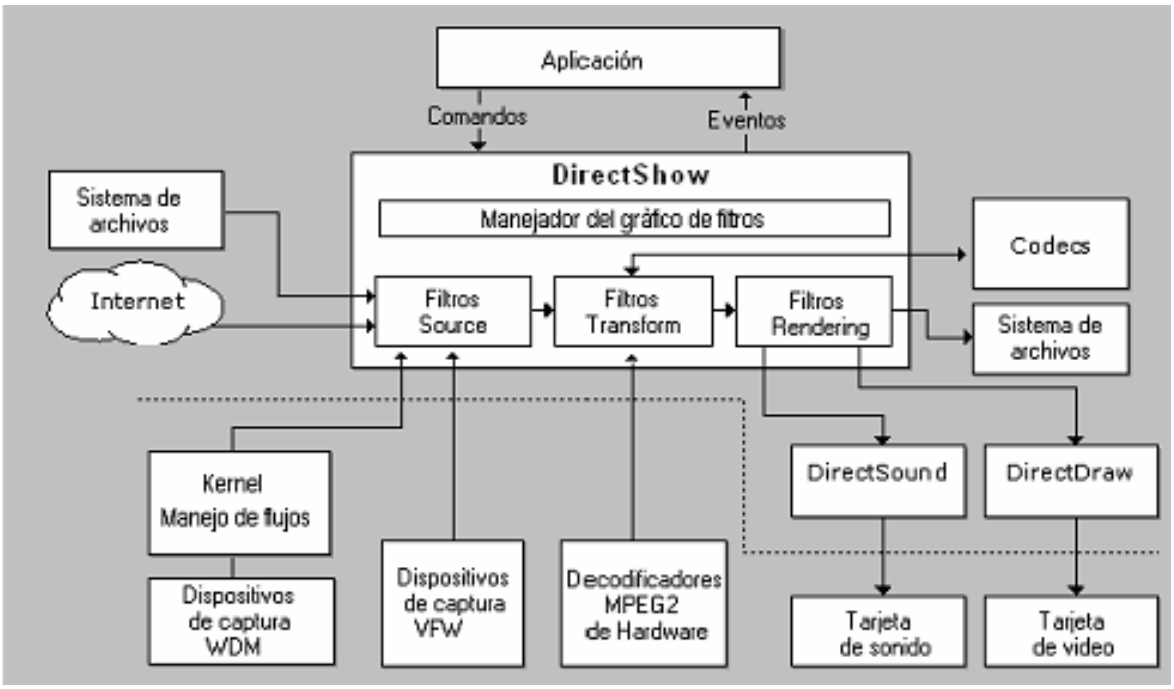


Figura IV. 15 Esquema global de DirectShow

DirectShow proporciona filtros para ejecutar archivos y flujos provenientes de varias fuentes, incluyendo archivos locales, CD, Internet, dispositivos DVD, sintonizadores de TV y tarjetas de captura de video.

DirectShow tiene compresores nativos, descompresores para algunos formatos de archivos, y muchos decodificadores de hardware y software compatibles con DirectShow. La reproducción hace completo uso de las capacidades de aceleración en hardware cuando este lo soporta.

4.5.1 El grafico de filtros y sus componentes

El Manejador del Gráfico de filtros provee un conjunto de interfaces COM para que las aplicaciones puedan acceder al gráfico de filtros. Así las aplicaciones pueden llamar a las interfaces del Manejador del Gráfico de filtros para controlar los flujos.

Un gráfico de filtros se compone de 3 tipos distintos de filtros:

- **Filtros Source**, toman datos provenientes de una fuente y los introducen a un gráfico de filtros.
- **Filtros Transform**, toma datos, los procesa y los envía.
- **Filtros Rendering**, proporciona los datos, a un dispositivo hardware para su presentación al usuario, pero puede enviarlos hacia cualquier localidad que acepte datos multimedia como memoria o disco.

Para que el gráfico de filtros trabaje, los filtros deben ser conectados en el orden apropiado y el flujo de datos comenzar y parar en dicho orden.

Controlar el flujo significa iniciar, pausar o detener el flujo multimedia. El Manejador del gráfico de filtros permite a la aplicación especificar estas actividades y llama a los métodos apropiados en los filtros. También la aplicación puede obtener el estado actual de los filtros, esta relación se puede apreciar en la Figura IV. 16.

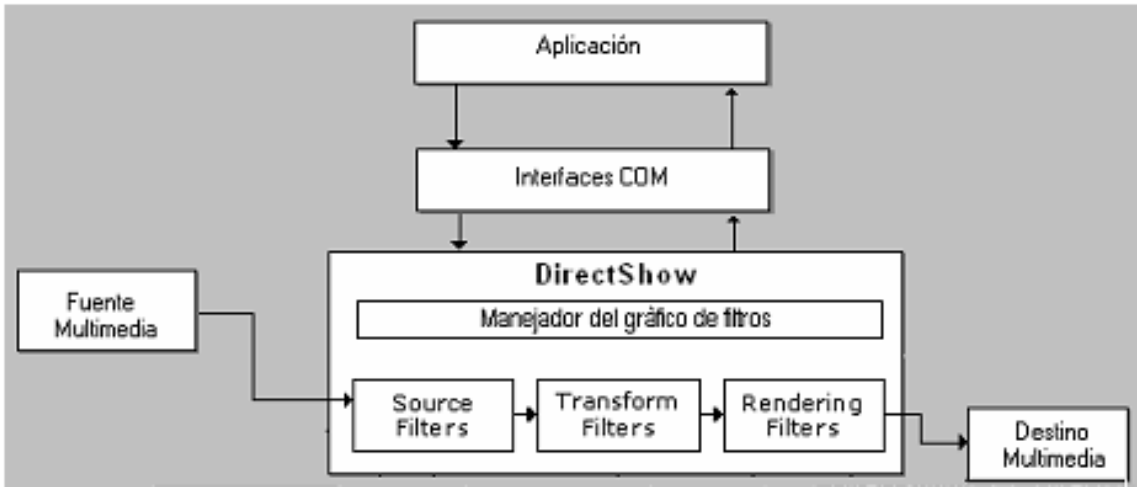


Figura IV. 16 Interacción del gráfico de filtros con la aplicación

Un filtro es un objeto COM que realiza una tarea específica. Para cada flujo se tiene al menos un pin. Un pin es un objeto COM creado por el filtro, que representa un punto de conexión para un flujo unidireccional de datos en el filtro.

4.5.2 Filtros

Los filtros son los bloques básicos de construcción en DirectShow. DirectShow separa el procesamiento de datos multimedia en una serie discreta de pasos, y un filtro representa un (o algunas veces más de uno) paso de procesamiento. Esto permite a las aplicaciones “mezclar y acoplar” filtros para realizar distintos tipos de operaciones en diferentes formatos multimedia y utilizando diversos dispositivos de hardware y software.

Por ejemplo el filtro Async File Source (Archivo fuente asíncrono) lee un archivo de un disco, el filtro TV Tuner (Sintonizador de TV) cambia el canal en una tarjeta de captura de TV, y el filtro MPEG-2 Splitter (Bifurcador MPEG-2) parsea los datos de audio y video en un flujo MPEG, así estos pueden ser decodificados. Aunque cada uno de estos filtros hace algo único internamente, desde el punto de vista de una aplicación, cada uno es solo un filtro DirectShow con ciertas características estándar, a saber: soporte para la interfaz IBaseFilter y uno o más pins de entrada y/o pins de salida que representan las conexiones a uno o más filtros DirectShow.

Todos los filtros caen en una de estas tres categorías: filtros source, filtros transform, y filtros renderer. A continuación se verá las características de cada uno de ellos.

4.5.2.1 Filtros Source

Los filtros source presentan los datos multimedia en bruto (En este contexto se entiende como datos sin procesamiento alguno provenientes quizás de un convertidor A/D con un formato de bytes sin codificar) para procesamiento. Ellos pueden obtenerlos de un archivo en un disco duro, o desde un CD o DVD, también

pueden obtenerse desde una fuente “live” tal como una tarjeta receptora de televisión o una tarjeta de captura conectada a una cámara digital. Algunos filtros source simplemente pasan los datos en bruto hacia un filtro parseador o un filtro splitter, mientras otros filtros también realizan el parseo.

4.5.2.2 Filtros Transform

Los filtros transform aceptan datos crudos o datos parcialmente procesados y realizan un procesamiento de dichos datos. Hay muchos tipos de filtros transform incluyendo parseadores que dividen los flujos de datos en bruto en muestras o frames (una imagen digitalizada compuesta por un arreglo cuadrado n x m de pixeles), filtros compresores, descompresores, y convertidores de formato.

4.5.2.3 Filtros Renderer

Los filtros renderer generalmente aceptan datos completamente procesados y ejecutan estos en el monitor o a través de los parlantes, o posiblemente a través de algún dispositivo externo. En esta categoría están incluidos los filtros “file-writer” que salvan datos a disco u otro elemento de almacenamiento. Los filtros renderer de video usan DirectDraw para desplegar video y el filtro renderer de audio usa DirectSound para ejecutar audio.

4.5.3 Pins

Los pins son responsables de proveer interfaces para conectarse con otros pins y para transportar los datos. Las interfaces de los pins soportan:

- La transferencia de datos entre filtros usando memoria compartida u otro método.

- La negociación del formato de datos para cada conexión pin a pin.
- La manipulación del buffer y negociación de su localización, con la intención de minimizar la copia de datos y maximizar su transferencia.

Las interfaces pin difieren poco dependiendo si son de entrada o salida.

Figura IV. 17 Conexión de pins

4.5.4 Muestras multimedia

Después que los datos en bruto han sido colocados en el gráfico, provenientes de un archivo local o de una tarjeta de captura, los bytes deben ser codificados en unidades básicas llamadas muestras multimedia (media samples). Algunas veces el filtro source hace la codificación y a veces un filtro aparte realiza dicha tarea. Una muestra multimedia es contenida por un objeto COM que implementa IMediaSample2. Adicionalmente a los datos multimedia actuales, el objeto contiene información incluyendo el tipo de media específico y los tiempos de sincronización.

Un objeto muestra multimedia que contenga datos de video, mantendrá los datos para un frame de video. Para audio, contiene los datos de varias muestras de sonido. En cualquier caso, cuando los datos se mueven desde el filtro source hacia el render (downstream) a través de un gráfico, se hace en la forma de objetos media sample. Downstream indica un flujo de datos “normal” desde el filtro que hace la adquisición de datos hacia el filtro que hace la presentación o almacenamiento de los mismos. En algunas ocasiones también se lo entiende como el filtro adyacente hacia el cual se dirige el flujo de datos.

4.5.5 Asignadores

Cuando dos filtros se conectan, sus pins deben ponerse de acuerdo en los detalles de cómo los objetos media sample serán transportados desde el filtro upstream (El término upstream indica un flujo de datos opuesto al downstream) hacia el filtro downstream. En este contexto “conectar” significa determinar el tamaño, localización y número de muestras que serán utilizadas. El tamaño de las muestras dependerá del tipo de media y el formato, la localización del buffer puede estar en memoria principal o en el dispositivo hardware tal como una tarjeta de captura de video. La creación y manipulación de las muestras se realiza por un asignador, este es un objeto COM usualmente creado por el pin de entrada en el filtro downstream. Para la mayoría de los casos, los detalles de la localización de los buffers serán completamente transparentes a las aplicaciones. Debe aclararse que el “movimiento” de datos en un gráfico de filtros no siempre involucra una operación de copia.

4.5.6 Relojes

En cualquier operación relacionada con multimedia, es vital sincronizar las muestras, así los frames de video serán desplegados a la razón apropiada, de modo que el flujo de audio no se adelantará al video ni lo opuesto. Un gráfico de filtros DirectShow contendrá exactamente un reloj que todos los filtros utilizarán como su base de tiempo. El Manejador del gráfico de filtros selecciona un reloj (o provee uno si es necesario) e informa a todos los filtros para que utilicen este reloj como fuente de sincronización.

4.6 FLUJO DE DATOS EN EL GRÁFICO DE FILTROS

4.6.1 Samples y Buffers

Cuando dos pins se conectan, los datos pueden moverse desde el pin salida hacia el pin entrada. El pin salida entrega datos, mientras el pin entrada recibe datos. La dirección del flujo de datos, desde el pin salida hacia el pin entrada se denomina downstream, y la dirección opuesta es llamada upstream.

El tipo de los datos que se desplazan entre dos pines depende de la implementación de los pines. En la mayoría de los casos, los pines trabajan con datos multimedia que se encuentran en la memoria principal. Sin embargo son posibles otras configuraciones. Por ejemplo, si dos filtros controlan una parte del hardware de video, el hardware quizás se encargue de los datos de video, con los pins intercambiando información de control. El tipo de datos, y como se mueven estos entre los pines, se denomina transporte (transport). Esta sección se avoca al caso donde los datos multimedia son contenidos en la memoria principal, llamada memoria local de transporte (local memory transport).

En la memoria local de transporte, los datos son empacados en objetos discretos llamados muestras multimedia (media samples). Una muestra multimedia es un objeto COM que mantiene un apuntador a un buffer de memoria. Una muestra multimedia soporta la interfaz ImediaSample.

Otro objeto COM es el asignador de memoria (memory asignator), responsable de ubicar buffers y crear muestras multimedia. En el tiempo de conexión, el asignador reserva memoria para los buffers. El asignador también crea un conjunto de

muestras multimedia y da a cada muestra multimedia un apuntador a una dirección dentro del bloque de memoria. Mientras el buffer no sea liberado, el asignador mantiene una lista de cuales muestras están disponibles. Cualquier filtro que necesite una muestra nueva, la solicita al asignador. Después de que la muestra es procesada, la muestra regresa a la lista.

Este mecanismo reduce la cantidad de memoria, debido a que los filtros re-utilizan los mismos buffers. Esto también previene a los filtros de la escritura accidental sobre los datos que no han sido procesados, debido a que el asignador mantiene la lista de las muestras disponibles. Finalmente este provee una manera eficiente de mover datos a través del gráfico. Cuando un pin salida entrega una muestra, este pasa un apuntador a la interfaz IMediaSample de la muestra. Esto provoca que no tenga que copiar ningún dato.

La Figura IV. 18 muestra la relación entre el asignador, las muestras multimedia, y el filtro.

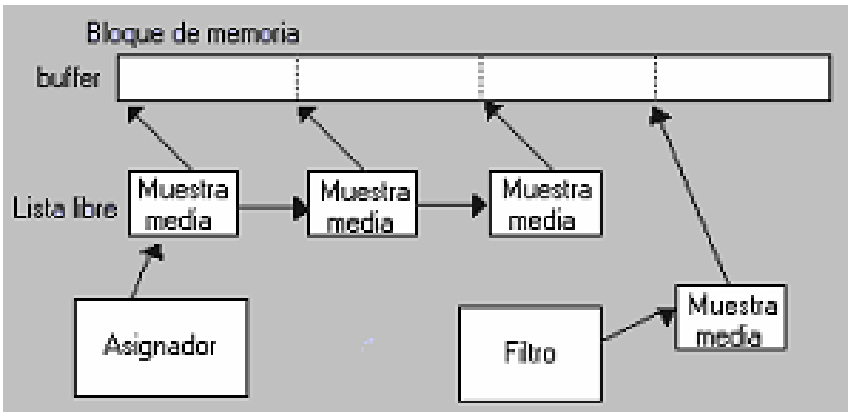


Figura IV. 18 Función del asignador

Los pins conectados comparten un asignador de memoria. Un filtro puede usar diferentes asignadores para sus entradas y salidas. Esto es típico cuando el filtro

expande los datos, como en el caso de un filtro descompresor. Si las salidas no son mayores que las entradas, el filtro puede procesar los datos en el mismo lugar, sin moverlos hacia un nuevo buffer. En este caso, dos o más conexiones de pines pueden compartir el mismo asignador.

4.6.2 Entrega de Muestras

Los pins entrada que soportan memoria local de transporte exponen la interfaz ImemInputPin. El pin salida entrega una muestra al llamar el método ImemInputPin::Receive en el pin entrada. El pin entrada realiza una de las siguientes funciones:

- Rechaza la muestra
- Se bloquea hasta que se finaliza el procesamiento de una muestra.
- Regresa inmediatamente y procesa la muestra en otro hilo (thread).

El método ImemInputPin::ReceiveCanBlock determina si el pin entrada puede bloquearse en la llamada a Receive. El pin salida puede llamar a este método para determinar una estrategia apropiada de hilado (threading). Algunos filtros crean un hilo de ejecución, así que ellos pueden entregar muestras en segundo plano mientras están haciendo otro trabajo. Otros filtros simplemente se bloquean hasta que el filtro downstream está listo para aceptar otra muestra.

Hay también un método para entregar más de una muestra a un tiempo, ImemInputPin::ReceiveMultiple. Este trabaja como Receive, pero con un arreglo de muestras.

4.6.3 Detener, Pausa, y Ejecución

Los filtros tienen tres estados: detenido, pausado y ejecutándose. Para muchos filtros, pausado y ejecutándose son equivalentes. Cuando un filtro está detenido, no se procesan muestras y se rechaza cualquier muestra de filtros upstream.

Cuando un filtro es pausado o está ejecutándose, este acepta muestras y las procesa. Si este es un filtro source, genera nuevas muestras y las entrega downstream. Los filtros renderer son una excepción. Cuando un filtro renderer va de detenido a pausado, no se completa la transición sino hasta que reciba muestras media. En este punto, el filtro mantiene la muestra y se bloquea en la llamada a Receive. Los renderers de video dibujan la muestra como un frame fijo. Los renderers de audio no presentan la muestra hasta que ellos estén ejecutándose. En cualquier caso, el renderer no acepta más muestras hasta que pase de pausado a ejecutándose.

El Manejador del gráfico de filtros controla el estado del gráfico de filtros como un conjunto. Las transiciones de estados válidas son entre detenido y pausado, y entre pausado y ejecutándose. Transiciones entre detenido y ejecutándose deben ser a través de pausado (Si se llama a ImediaControl::Run en un gráfico detenido, o ImediaControl::Stop en un filtro ejecutándose, el Manejador del gráfico de filtros pausa primero el gráfico).

Cuando el grafico del filtro va de detenido hacia pausado, ocurre la siguiente secuencia:

El Manejador del gráfico de filtros llama a ImediaFilter::Pause en cada filtro, iniciando desde el filtro renderer y continuando upstream.

Como cada filtro conmuta a pausado, un filtro se encuentra listo para aceptar muestras desde su upstream más cercano. Este puede entregar muestras downstream con seguridad, debido a que los filtros downstream también están pausados.

El filtro source es el último en ser pausado. En este punto, el filtro inicia entregando muestras. Como las muestras se mueven downstream, los filtros intermedios las procesan. Cuando las primeras muestras alcanzan al renderer, el renderer completa su estado de transición y se bloquea.

Estos eventos pueden tomar una cantidad arbitraria de tiempo para completarse (usualmente no mucho, pero el retardo podría ser significativo, especialmente si la fuente requiere descompresión). Al tiempo que el gráfico ha terminado de pausarse, estará apto para iniciar la presentación de datos inmediatamente. Una aplicación puede hacer esto adelantándose en tiempo, y entonces rápidamente conmutar el grafico a ejecutándose, en respuesta a un comando del usuario.

4.6.4 Notificación de eventos en DirectShow

Un filtro notifica al Manejador del gráfico de filtros acerca de un evento mediante el envío de una notificación de evento. El evento puede ser algunas veces previsto, tal como el fin de un flujo, o este podría representar un error, tal como una falla al presentar un flujo. El Manejador del gráfico de filtros maneja algunos eventos de los filtros y deja otros eventos para que los maneje la aplicación. Si el Manejador del

gráfico de filtros no maneja el evento del filtro, este coloca la notificación del evento en una cola. El Gráfico de filtros puede también encolar sus propias notificaciones de eventos para la aplicación.

Una aplicación obtiene eventos de la cola y responde a ellos basado en el tipo de evento. La notificación de eventos en DirectShow es similar al esquema de encolado de mensajes de Windows. Una aplicación puede también cancelar el comportamiento por default del Manejador del gráfico de filtros para un tipo de evento dado. El Manejador del gráfico de filtros entonces mete estos eventos directamente en la cola para que la aplicación las maneje.

Este mecanismo permite:

- Al Manejador del gráfico de filtros comunicarse con la aplicación.
- A los filtros comunicarse con la aplicación y con el Manejador del Gráfico de filtros.
- A la aplicación determinar su grado de involucramiento en el manejo de eventos.

4.6.4.1 Capturando Eventos

El Manejador del gráfico de filtros expone tres interfaces que soportan la notificación de eventos.

- **IMediaEventSink** contiene los métodos para que los filtros envíen eventos.
- **IMediaEvent** contiene métodos para que las aplicaciones capturen eventos.
- **IMediaEventEx** hereda desde y extiende la interfaz IMediaEvent.

Los filtros envían notificaciones de eventos mediante la llamada al método IMediaEventSink::Notify en el Manejador del Gráfico de filtros, y dos parámetros DWORD que dan información adicional. Dependiendo del código del evento, los parámetros pueden contener apuntadores, regresar códigos, tiempos de referencia, u otra información.

Para obtener un evento de una cola la aplicación llama al método IMediaEvent::GetEvent en el Manejador del Gráfico de filtros. Este método se bloquea hasta que regrese un evento o hasta que pase un lapso de tiempo especificado. Asumiendo que hay un evento encolado, el método regresa con el código del evento y los dos parámetros del evento. Después de llamar a GetEvent, una aplicación debe siempre llamar al método IMediaEvent::FreeEventParams para actualizar cualquier recurso asociado con los parámetros del evento. Por ejemplo, un parámetro podría ser un valor BSTR que fuese localizado por el Gráfico de filtros.

El siguiente código provee una guía de cómo obtener eventos de una cola.

```
long evCode, param1, param2;
HRESULT hr;
while (hr = pEvent->GetEvent(&evCode, &param1, &param2, 0), SUCCEEDED(hr))
{
   switch(evCode)
   {
      // Llama a funciones definidas en la aplicación para cada
      // tipo de evento que se quiera manejar.
}

   hr = pEvent->FreeEventParams(evCode, param1, param2);
}
```

Para sobrescribir el manejador por default del Manejador del Gráfico de filtros, se debe llamar al método IMediaEvent::CancelDefaultHandling con el código del evento como un parámetro. Se puede volver a crear una instancia del manejador por default llamando al método IMediaEvent::RestoreDefaultHandling.

4.6.4.2 Saber cuándo un evento ocurre

Adicionalmente a lo descrito, una aplicación necesita una manera de saber cuándo los eventos están esperando en la cola. El Manejador del gráfico de filtros provee dos formas de hacer esto.

- Usando notificaciones Windows, el Manejador del gráfico de filtros envía un mensaje tipo Windows a una ventana de aplicación en cualquier momento que exista un nuevo evento.
- Usando manejadores de eventos, la aplicación obtiene un manejador a un evento Windows manual-reset. El Manejador del gráfico de filtros señala el evento manual-reset cuando haya notificaciones de eventos en la cola y reinicializa esta cuando la cola está vacía.

4.6.4.3 Notificación Windows

Para utilizar una notificación windows, se debe llamar al método IMediaEventEx::SetNotifyWindow y especificar un mensaje privado. Las aplicaciones pueden usar números de mensajes en el rango de WM_APP hasta 0xBFFF como mensajes privados. En cualquier momento el Manejador del gráfico de filtros coloca una nueva notificación de evento en la cola, este envía este mensaje a la ventana

destino. La aplicación responde al mensaje desde el lazo (loop) de mensajes de la ventana. El siguiente código ejemplifica como ajustar la notificación window.

```
#define WM_GRAPHNOTIFY WM_APP + 1   // Mensaje privado.
pEvent->SetNotifyWindow((OAHWND)g_hwnd, WM_GRAPHNOTIFY, 0);
```

Este mensaje es un mensaje ordinario Windows, y es enviado separadamente de la cola de notificación de eventos DirectShow. La ventaja de esta aproximación es que muchas aplicaciones ya tienen implementado un lazo de mensajes. Así mismo puede incorporar un manejador de eventos DirectShow sin mucho trabajo adicional.

El siguiente código muestra cómo responder al mensaje de notificación.

```
LRESULT CALLBACK WindowProc( HWND hwnd, UINT msg, UINT wParam, LONG lParam)
{
   switch (msg)
   {
case WM_GRAPHNOTIFY:
HandleEvent();  // Función definida en la aplicación.
          break;

      // Maneja otros mensajes de Windows aquí también.
}
   return (DefWindowProc(hwnd, msg, wParam, lParam));
}
```

Debido a que la notificación de eventos, y el lazo de mensajes son ambos asíncronos, la cola puede contener más de un evento para el tiempo que su aplicación responda al mensaje. También, los eventos pueden algunas veces ser quitados de la cola si ellos fuesen inválidos. Así mismo, en el código manejador del

evento, hay que llamar a GetEvent hasta que este regrese un código de falla, indicando que la cola está vacía.

4.6.4.4 Manejadores de eventos

El gráfico de filtros guarda un evento manual-reset que refleja el estado de la cola de eventos. Si la cola contiene notificaciones de eventos pendientes, el gráfico de filtros señala los eventos manual-reset. Si la cola está vacía, una llamada al método MediaEvent::GetEvent reinicializa el evento. Una aplicación puede usar este evento para determinar el estado de la cola.

El método IMediaEvent::GetEventHandle obtiene un manejador a el evento manual-reset. Espera para que el evento sea señalado mediante un llamado a una función tal como WaitForMultipleObjects. Al mismo tiempo que el evento es señalado, la llamada al evento IMediaEvent::GetEvent obtiene la notificación del evento.

El siguiente código de ejemplo ilustra este método. Este obtiene el manejador de evento, entonces espera en intervalos de 100 milisegundos para ser señalado. Si el evento es señalado, este llama a GetEvent e imprime el código de evento y los parámetros del evento a la ventana. El lazo termina cuando ocurre el evento EC_COMPLETE.

```
HANDLE  hEvent;
long    evCode, param1, param2;
BOOLEAN bDone = FALSE;
HRESULT hr = S_OK;

hr = pEvent->GetEventHandle((OAEVENT*)&hEv);

while(!bDone)
```

```
{
  if (WAIT_OBJECT_0 == WaitForSingleObject(hEvent, 100))
   {
     while (hr = pEvent->GetEvent(&evCode, &param1, &param2, 0), SUCCEEDED(hr))
      {
         printf("Event code: %#04x\n    Params: %d, %d\n", evCode, param1, param2);
         hr = pEvent->FreeEventParams(evCode, param1, param2);
bDone = (EC_COMPLETE == evCode);
      }
   }
}
```

Debido a que el gráfico de filtros automáticamente ajusta o reinicialaza el evento cuando sea apropiado, la aplicación no debería hacerlo. También, cuando se actualiza el gráfico de filtros, el gráfico de filtros cierra el manejador de eventos, así no hace uso del manejador de eventos después de este punto.

4.7 HARDWARE EN EL GRÁFICO DE FILTROS

4.7.1 Filtros Envoltura (Wrapper)

Todos los filtros DirectShow son componentes de software en el modo usuario. Para que un dispositivo hardware en el modo kernel como una tarjeta de captura de video/sonido se añada al gráfico de filtros de DirectShow, el dispositivo debe ser representado como un filtro en el modo usuario dentro del gráfico de filtros. Esta función se lleva a cabo para varios tipos de dispositivos por filtros especializados llamados filtros envoltura (wrapper filters), provistos con DirectShow.

Los filtros envoltura trabajan soportando las interfaces COM que representan las capacidades esperadas del dispositivo. La aplicación usa estas interfaces para

pasar información hacia y desde el filtro, el filtro traduce las llamadas a la interface en información que el controlador del dispositivo pueda comprender, y entonces el filtro pasa esta información hacia y desde el controlador en el modo kernel.

Para los desarrolladores de aplicaciones, el principio de envolver dispositivos de hardware como filtros en modo usuario significa que las aplicaciones controlan dispositivos de la misma forma que ellos controlan cualquier otro filtro DirectShow. No se requiere una programación especial en la parte de la aplicación; todos los detalles involucrados en la comunicación con el dispositivo en el modo kernel son encapsulados dentro del filtro mismo.

4.7.2 Video para dispositivos Windows

Para soportar las últimas tarjetas de captura Video For Windows, DirectShow provee el filtro de captura VFW. Cuando una tarjeta VfW está presente en el sistema, esta puede ser descubierta usando el System Device Enumerator (Aplicación contenida en DirectX 11.0). El Enumerator se utiliza también para añadir el filtro de captura VfW al gráfico y asociar este con la tarjeta que fue encontrada. El resto del gráfico de filtros puede entonces ser construido de la manera usual.

4.7.3 Captura de Audio y dispositivos Mezcladores

Las tarjetas de sonido modernas tienen conectores de entrada para micrófonos y otro tipo de dispositivos. Estas tarjetas también tienen capacidades de mezclado para controlar el volumen, graves y agudos de cada entrada individualmente. En DirectShow, las entradas de las tarjetas de sonido y mezcladoras son envueltas en el filtro de captura de audio (Audio Capture Filter). Cada tarjeta de sonido en el

sistema local puede ser descubierta con el system device enumerator. Para visualizar las tarjetas de sonido en el sistema basta abrir el programa GraphEdit que se instala con el SDK y seleccionar la categoría Audio Capture Sources. Cada filtro representado aquí es una instancia separada del Audio Capture Filter.

CAPÍTULO V

DISEÑO E IMPLEMENTACIÓN

5.1 INTRODUCCIÓN

Para la ejecución de este proyecto, se realizó una investigación de ideas similares, incluyendo las propuestas de uso comercial. La principal herramienta para la búsqueda fue el Internet.

Se encontró trabajos que tienen algunos puntos en común con la idea planteada, pero en ningún caso cubren la totalidad de usos, aplicaciones y la apertura que se proponen para la realización del proyecto.

El punto de partida para este proyecto es DirectX.

DirectX ofrece funcionalidades útiles multimedia que se pueden utilizar para el desarrollo de software que hacen uso de gráficos, audio y/o video. Para acceder a todas las capacidades multimedia de Windows. Gran cantidad de software se han desarrollado utilizando las funciones de DirectX, debido a los pocos cambios de estructura que se han producido en el transcurso de los años.

La forma normal de uso y funcionalidad de DirectX y DirectShow es usar el lenguaje de programación C++, con el uso de DirectXSDK con sus respectivas librerías. Ejemplos de software libre, que utilizan este tipo de programación son AMCap, VidCap, Dscaler, todos ellos de código muy extenso y difícil de comprender.

Una gran opción para los desarrolladores de software es “The Code Project”. Este sitio web se creó para ofrecer un lugar de reunión e intercambio de ideas, ofreciendo a los desarrolladores todos los recursos que necesitan para ayudarles en su día a día de programación.

The Code Project es sin lugar a dudas uno de los más completos e interesantes sitios sobre programación para Windows existentes en la Red. Esencialmente es un portal formado por una comunidad enorme de miembros (que superan actualmente los dos millones y medio) en la que todo el mundo puede aportar sus conocimientos en diversas áreas de desarrollo: C++, C#, MFC, ASP, ASP .NET y la plataforma .NET en general.

Este proyecto se basa en el trabajo desarrollado por Brian Low, publicado en el sitio web The Code Project, el cual ofrece varias funciones para la captura de audio y video, escrito en lenguaje de programación C#. Algunas características que tiene

este programa son: soporte para tarjetas capturadoras de tv, grabación de video, uso de códecs de compresión, entre otros.

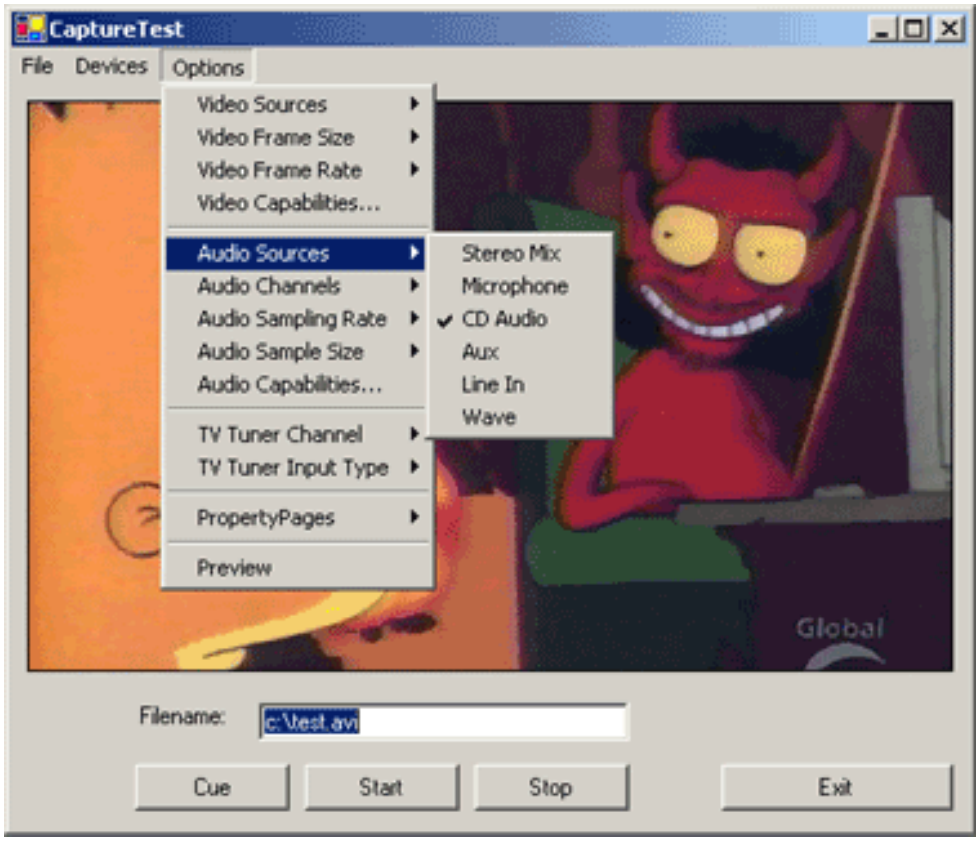


Figura V. 19 Vista principal del programa desarrollado por Brian Low

5.2 DESCRIPCIÓN DEL CÓDIGO

5.2.1 DirectShowNET

El objetivo de esta biblioteca es permitir el acceso a las funcionalidades de Microsoft DirectShow desde .NET. Esta biblioteca es compatible con Visual Basic .NET y C #, y, en teoría, debería funcionar con cualquier lenguaje .NET.

Esta biblioteca ofrece las enumeraciones, estructuras y definiciones de interfaz para acceder a ellas.

5.2.2.1 Interfaces

Son funciones que permiten el acceso a las interfaces de DirectShow. Una interface es un conjunto de métodos (funciones) y atributos (datos) que tienen una fuerte relación lógica entre ellos.

- **DsControl**

  → publicinterfaceIMediaControl
  Proporciona métodos para controlar el flujo de datos a través del gráfico de filtro.

  → publicinterfaceIMediaEvent
  Contiene métodos para recibir las notificaciones de eventos y para reemplazar el manejo por defecto del administrador de graficos de filtros.

  → publicinterfaceIMediaEventEx
  Añade métodos que le permiten a la aplicación recibir mensajes cuando ocurren los eventos.

  → publicinterfaceIBasicVideo2
  La interfaz IBasicVideo2 es una extención de IBasicVideo. Los filtros Video Renderer y Video Mixing Renderer implementan esta interfaz, pero la interfaz se expone a la aplicacion a través del administrador de gráfico de filtros.

  → publicinterfaceIVideoWindow
  Establece las propiedades en la ventana de vídeo, tales como el propietario, estilo, la posición y dimension de la ventana entre otras.

  → publicinterfaceIMediaPosition
  Establecen propiedades como la duración del flujo de video, las horas de inicio y parada, la tasa y la posición actual.

→ publicinterfaceIBasicAudio
Controla el volumen y el balance del flujo de audio.

- **Dscore**

→ publicinterfaceIPin
Esta interfaz representa un solo punto unidireccional de la conexión con un filtro. Un pin se conecta con exactamente un pin en otro filtro. Otros objetos pueden utilizar la interfaz en este pin.

→ publicinterfaceIFilterGraph
La interfaz IFilterGraph proporciona métodos para construir un gráfico de filtro. La aplicación puede utilizarla para agregar filtros al gráfico, conectar o desconectar los filtros, eliminarlos y realizar otras operaciones básicas.

→ publicinterfaceIPersist
Proporciona el CLSID del objeto que puede almacenarse persistentemente en el sistema. Permite al objeto especificar qué controlador de objetos usar en el proceso del cliente, ya que se utiliza en la implementación predeterminada de cálculo de referencias.

→ publicinterfaceIMediaFilter
Controla el estado de transmisión de un filtro. Proporciona métodos para la conmutación entre los estados de los filtros (parada, pausa y ejecución); recupera el estado actual del filtro y puede establecer un reloj de referencia.

→ publicinterfaceIBaseFilter
Es la interfaz principal de los filtros DirectShow. El administrador de gráfico de filtros utiliza esta interfaz para controlar los filtros.

→ publicinterfaceIEnumPins
La aplicación utiliza esta interfaz para enumerar los pines del filtro.

→ publicinterfaceIMediaSample
Obtiene las propiedades de las muestras multimedia. Una muestra multimedia es un objeto COM que contiene un bloque de datos de multimedia.

- **DsExtend**

→ publicinterfaceICaptureGraphBuilder2
Proporciona métodos para la construcción de gráficos de captura, y otros gráficos de filtros personalizados.

→ publicinterfaceIGraphBuilder
Proporciona métodos que permiten a la aplicación crear un gráfico de filtro.

→ publicinterfaceIFileSinkFilter2
Guarda la salida del filtro de compresión de vídeo en un archivo.

→ publicinterfaceIVideoFrameStep
Esta interfaz permite mostrar una secuencia de vídeo tan lentamente como un cuadro a la vez.

→ publicinterfaceIAMStreamConfig
Establece el formato de salida en determinadas capturas y filtros de compresión, tanto de audio como de vídeo.

→ publicinterfaceIAMTuner
Se utiliza para controlar el sintonizador de TV con Microsoft DirectShow.

→ publicinterfaceIAMVideoProcAmp
Permite a la aplicación ajustar las características de la señal de vídeo entrante, como el brillo, contraste, tono, saturación, gamma, y la nitidez.

→ publicinterfaceIAMAnalogVideoDecoder
Establece y recupera información sobre el proceso de conversión de analógico a digital de un filtro de captura de vídeo.

→ publicinterfaceIAMTVTuner
Controla el sintonizador de TV. El filtro del sintonizador de TV implementa esta interfaz. La aplicación puede utilizarla para seleccionar los canales de televisión, para obtener información acerca de sus frecuencias y para determinar cuáles son las normas de vídeo analógicas de la tarjeta de sintonizador de TV compatible.

→ publicinterfaceIAMVfwCompressDialogs
Establece un cuadro de diálogo con el códec de video for Windows (VFW), generalmente para configurar o recuperar el estado del compresor.

→ publicinterfaceIAMVideoCompression
Establece y obtiene las propiedades de compresión de vídeo.

→ publicenumAMTunerModeType
Especifica la frecuencia de un sintonizador de TV (cable o antena).

→ publicenumAMTunerEventType
Especifica el estatus de cambio de canal.

→ publicenumAnalogVideoStandard
Especifica el estándar del vídeo analógico (NTSC, NTSC-M, PAL, SECAM, etc)

→ publicinterfaceISpecifyPropertyPages
Indica que un objeto admite página de propiedades.

→ publicinterfaceIVMRFilterConfig9
La aplicación utiliza esta interfaz para configurar el modo de funcionamiento de VMR (Video Mixing Renderer) y los mecanismos de procesamiento de vídeo.

→ publicinterfaceIVMRWindowlessControl9
Controla el filtro Video Mixing Renderer9 (VMR-9), representa un flujo de vídeo dentro de una ventana de contenedores.

→ publicinterfaceIVMRWindowlessControl
Controla el filtro Video Mixing Renderer7 (VMR-7), representa un flujo de vídeo dentro de una ventana de contenedores.

→ publicinterfaceIVMRFilterConfig
Se utiliza para configurar el modo de funcionamiento y los mecanismos de vídeo del filtro Video Mixing Renderer7 (VMR-7).

→ publicinterfaceISampleGrabber
Permite a la aplicación recuperar muestras multimedias a medida que se mueven a través del grafico de filtros.

5.2.2 DirectX.Capture

Es una biblioteca de clases para la captura de audio y vídeo a los archivos AVI en formato .NET. Algunas de las tareas que permite realizar esta biblioteca son: Enlistar y seleccionar los dispositivos de hardware, acceder al audio del dispositivo hardware y a la configuración de vídeo (por ejemplo, frame rate y frame size), Soporte de codecs de compresión de audio y vídeo, previsualización de vídeo, soporte de sintonizadores de TV, mostrar las propiedades de los dispositivos hardware.

5.2.2.1 Clases

→ publicclassFilter : IComparable
Representa un filtro de DirectShow, por ejemplo el dispositivo de captura de vídeo, el codec de compresión.

→ publicclassDxUtils: CollectionBase
Son los espacios de color añadidos a los GUID en la propiedad Color Space, por ejemplo AYUV, UYVY, Y411, Y41P, Y211, etc.

→ publicclassPropertyPage : IDisposable
Es una clase base para la representación de las propiedades de páginas expuestas por los filtros, por ejemplo la propiedad tv tuner del sintonizador de tv.

→ publicclassSource : IDisposable
Representa una conexión física o un dispositivo de audio/video conectado a la computadora.

→ publicclassTuner : IDisposable
Realiza el control de un dispositivo sintonizador de TV.

→ publicclassVideoCapabilities
Muestra las propiedades del dispositivo de vídeo, tales como tamaño del marco mínimo/máximo y velocidad de fotogramas.

→ publicclassCapture : ISampleGrabberCB
Captura el audio y vídeo en archivos AVI. La clase capture sólo requiere un dispositivo de audio y/o video para iniciar la captura.

5.2.2.2 Descripción de los métodos (funciones) de las clases más utilizadas

Clase Capture

La clase capture se encarga de todo lo referente a la captura de audio y video, modificación de sus caracterisitcas básicas (framerate, framesize)

→ publicenumRecFileModeType
Enumera los formatos de grabación como .avi, .wmv, .wma.

→ publicstring Filename
Nombre del archivo con el que se guarda la captura.

→ publicRecFileModeType RecFileMode
Controla el modo de grabación de los archivos establecidos en el método RecFileModeType.

→ publicControl PreviewWindow
Controla si se muestra o no en pantalla la captura del video.

→ publicVideoCapabilities VideoCaps
Contiene la información de la configuración del video (frame rate, frame size).

→ publicAudioCapabilities AudioCaps
Contiene la información de la configuración del audio

→ publicFilter VideoDevice
Contiene la información de los dispositivos de captura de video.

→ publicFilter AudioDevice
Contiene la información de los dispositivos de captura de audio.

→ publicFilter VideoCompressor
Muestra los códecs de compresión de video instalados en el sistema.

→ publicFilter AudioCompressor
Muestra los códecs de compresión de audio instalados en el sistema.

→ publicSource VideoSource
Indica los tipos de entrada de las tarjetas presentes como s-video, tv tuner.

→ publicSource AudioSource
Indica los tipos de entrada de las tarjetas presentes LR, audio tuner.

→ publicTuner Tuner
Indica si el dispositivo de video actual tiene o no un sintonizador de TV, si lo tiene muestra las propiedades de este (Channel, Country Code, etc).

→ publicdouble FrameRate
Obtiene y establece la velocidad de fotogramas para la captura de video.

→ publicSize FrameSize
Obtiene y establece el tamaño del marco utilizado para la captura de vídeo.

→ publicshort AudioChannels
Obtiene o establece el número de canales del audio (mono o estéreo).

→ publicint AudioSamplingRate
Obtiene o establece el número de muestras de audio tomadas por segundo. Tipos comunes de muestreo son 8 KHz, 11.025 KHz, 22.05 KHz y 44.1 KHz. No todas las frecuencias de muestreo son compatibles.

→ publicshort AudioSampleSize
Obtiene o establece el número de bits de grabación por muestra. Los tamaños más comunes de la muestra son de 8 bitsy 16 bits.

→ publicbool AudioAvailable
Compruebe si hay un dispositivo de audio.

→ publicvoid Start
Empieza la captura.

→ publicvoid Stop
Detiene la captura actual.

→ publicbool UseVMR9
Comprueba sise debe utilizarVMR9.

→ publicbool ShowPropertyPage
Muestra las propiedades del dispositivo de video de entrada

→ publicVideoCapabilities PreviewCaps
Contiene la información de la configuración del video (frame rate, frame size).

→ protectedvoid createGraph()
Crea un nuevo gráfico de filtro y agrega filtros (dispositivos, compresores, misceláneos), pero no conecta los filtros. Utiliza renderGraph() para conectarlos filtros.

→ protectedvoid renderGraph()
Conecta los filtros del gráfico de filtros previamente creados por createGraph().

→ protectedenumGraphState
Posibles estados del grafico de filtros (Null, Created, Rendered, Capturing).

→ protectedvoid derenderGraph()
Desconecta y retira todos los filtros, excepto el dispositivo y los filtros de compresión. Esto es lo contrario de renderGraph(). Algunas propiedades, como FrameRate sólo se pueden establecer cuando los pines de salida del dispositivo no están conectados.

5.2.2.3 Proceso para capturar el video

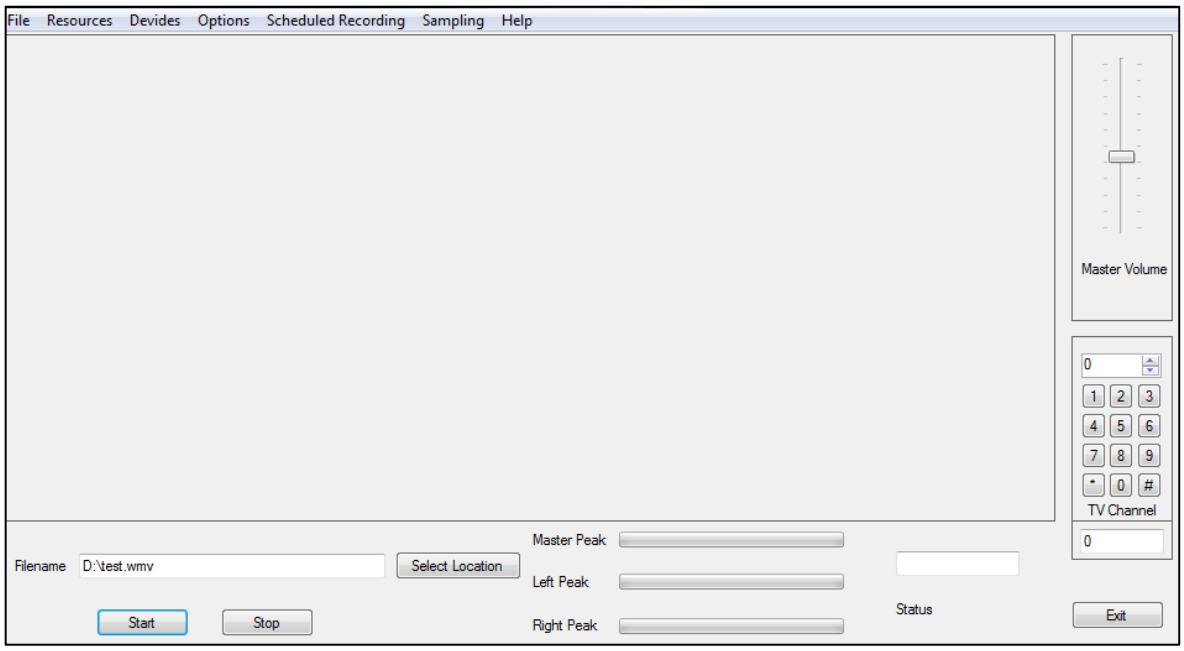


Figura V. 20 Pantalla principal aplicación TVisor

Inicia la carga de los menús llamando a las funciones initMenu() y updateMenu()

Función updateMenu

→ **Menú Resources**

Añade al Menú de Resources todos los dispositivos hardware de captura de vídeo presentes en el sistema (Web Cam, Tarjetas capturadoras TV, etc)

```
// Load resources
Filter resources = null;
if (capture != null)
    resources = capture.VideoDevice;
mnu_resources.MenuItems.Clear();
m = new MenuItem("(None)", new EventHandler(mnu_resources_Click));
```

```
m.Checked = (resources == null);
mnu_resources.MenuItems.Add(m);
for (int c = 0; c < filters.VideoInputDevices.Count; c++)
{
    f = filters.VideoInputDevices[c];
    m = new MenuItem(f.Name, new EventHandler(mnu_resources_Click));
    m.Checked = (resources == f);
    mnu_resources.MenuItems.Add(m);
}
mnu_resources.Enabled = (filters.VideoInputDevices.Count > 0);
```

→ **Menú Devides**

- **SubMenú A**udio Devices

Añade al Menú de Devices, submenú “Audio Devices” todos los dispositivos hardware de captura de audio presentes en el sistema (Dispositivos conectados a las entradas de la tarjeta de audio como el micrófono)

```
// Load audio devices
Filter audioDevice = null;
if (capture != null)
    audioDevice = capture.AudioDevice;
mnuAudioDevices.MenuItems.Clear();
m = new MenuItem("(None)", new EventHandler(mnuAudioDevices_Click));
m.Checked = (audioDevice == null);
mnuAudioDevices.MenuItems.Add(m);
```

```
for (int c = 0; c < filters.AudioInputDevices.Count; c++)
{
    f = filters.AudioInputDevices[c];
    m = new MenuItem(f.Name, new EventHandler(mnuAudioDevices_Click));
    m.Checked = (audioDevice == f);
    mnuAudioDevices.MenuItems.Add(m);
}
mnuAudioDevices.Enabled = (filters.AudioInputDevices.Count > 0);
```

- **SubMenú Vídeo Compressors**

Añade al Menú de Devices, submenú “Vídeo Compressors” los códecs de compresión de video instalados en el sistema. (Divx, Xvid, etc)

```
// Load video compressors
try
{
    mnuVideoCompressors.MenuItems.Clear();
    m = new MenuItem("(None)",new EventHandler(mnuVideoCompressors_Click));
    m.Checked = (capture.VideoCompressor == null);
    mnuVideoCompressors.MenuItems.Add(m);
    for (int c = 0; c < filters.VideoCompressors.Count; c++)
    {
        f = filters.VideoCompressors[c];
     m = new MenuItem(f.Name, new EventHandler(mnuVideoCompressors_Click));
        m.Checked = (capture.VideoCompressor == f);
```

```
        mnuVideoCompressors.MenuItems.Add(m);
    }
    mnuVideoCompressors.Enabled = ((capture.VideoDevice != null) &&
                                   (filters.VideoCompressors.Count > 0));
}
catch { mnuVideoCompressors.Enabled = false; }
```

- **SubMenú Audio Compressors**

Añade al Menú de Devices, submenú “Audio Compressors” los códecs de compresión de audio instalados en el sistema. (MPEG Layer-3, PCM, etc)

```
// Load audio compressors
try
{
    mnuAudioCompressors.MenuItems.Clear();
    m = new MenuItem("(None)", new EventHandler(mnuAudioCompressors_Click));
    m.Checked = (capture.AudioCompressor == null);
    mnuAudioCompressors.MenuItems.Add(m);
    for (int c = 0; c < filters.AudioCompressors.Count; c++)
    {
        f = filters.AudioCompressors[c];
        m = new MenuItem(f.Name, new EventHandler(mnuAudioCompressors_Click));
        m.Checked = (capture.AudioCompressor == f);
        mnuAudioCompressors.MenuItems.Add(m);
    }
```

```
    mnuAudioCompressors.Enabled = ((this.capture.AudioAvailable) &&
(this.filters.AudioCompressors.Count > 0));
}
catch { mnuAudioCompressors.Enabled = false; }
```

→ **Menú Options**

- **SubMenú Vídeo Standard**

Añade al menú options submenu “Video Standard” los estándares de Televisión que se puede utilizar. (NTSC-M, PAL, etc)

```
// Load video standards
menuVideoStandard1.MenuItems.Clear();
if ((this.capture != null) &&
            (this.capture.dxUtils != null) && (this.capture.dxUtils.VideoDecoderAvail))
{
    try
    {
        menuVideoStandard1.MenuItems.Clear();
        AnalogVideoStandard currentStandard = capture.dxUtils.VideoStandard;
        AnalogVideoStandard availableStandards =
                        capture.dxUtils.AvailableVideoStandards;
        int mask = 1;
        while (mask <= (int)AnalogVideoStandard.PAL_N_COMBO)
        {
            int avs = mask & (int)availableStandards;
```

```
            if (avs != 0)
            {
                m = new MenuItem(((AnalogVideoStandard)avs).ToString(), new
                        EventHandler(menuVideoStandard1_Click));
                m.Checked = (currentStandard == (AnalogVideoStandard)avs);
                menuVideoStandard1.MenuItems.Add(m);
            }
            mask *= 2;
        }
        menuVideoStandard1.Enabled = true;
    }
    catch { menuVideoStandard1.Enabled = false; }
}
else
{
    menuVideoStandard1.Enabled = false;
}
```

- **SubMenú Vídeo Sources**

Añade al Menú de Devices, submenú “Vídeo Sources”, los tipos de entrada de las tarjetas presentes. (s-vídeo, tv-tuner)

```
// Load video sources
try
{
```

```
    mnuVideoSources.MenuItems.Clear();
    capture.VideoSources = null;
    current = capture.VideoSource;
    for (int c = 0; c < capture.VideoSources.Count; c++)
    {
        s = capture.VideoSources[c];
        m = new MenuItem(s.Name, new EventHandler(mnuVideoSources_Click));
        m.Checked = (current == s);
        mnuVideoSources.MenuItems.Add(m);
    }
    mnuVideoSources.Enabled = (capture.VideoSources.Count > 0);
    if (current != null)
    {
        capture.VideoSource = current;
}
}
catch { mnuVideoSources.Enabled = false; }
```

- **SubMenú Audio** Sources

Añade al Menú de Devices, submenú “Audio Sources” los tipos de entrada de las tarjetas presentes. (LR, Audio Tuner)

```
// Load audio sources
try
{
```

```
    mnuAudioSources.MenuItems.Clear();
    this.capture.AudioSources = null;
    current = capture.AudioSource;
    for (int c = 0; c < capture.AudioSources.Count; c++)
    {
        s = capture.AudioSources[c];
        m = new MenuItem(s.Name, new EventHandler(mnuAudioSources_Click));
        m.Checked = (current == s);
        mnuAudioSources.MenuItems.Add(m);
    }
    mnuAudioSources.Enabled = (capture.AudioSources.Count > 0);
}
catch { mnuAudioSources.Enabled = false; }
```

- **SubMenú Frame Rate**

Añade al menú options submenú Video Capture Settings opción “Vídeo Frame Rate”, la velocidad de los fotogramas para la captura de vídeo.

```
// Load frame rates
try
{
    mnuFrameRates.MenuItems.Clear();
    int frameRate = (int)(capture.FrameRate * 1000);
    m = new MenuItem("15 fps", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 15000);
```

```
    mnuFrameRates.MenuItems.Add(m);
    m = new MenuItem("24 fps (Film)", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 24000);
    mnuFrameRates.MenuItems.Add(m);
    m = new MenuItem("25 fps (PAL)", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 25000);
    mnuFrameRates.MenuItems.Add(m);
    m = new MenuItem("29.997 fps (NTSC)", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 29997);
    mnuFrameRates.MenuItems.Add(m);
    m = new MenuItem("30 fps (~NTSC)", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 30000);
    mnuFrameRates.MenuItems.Add(m);
    m = new MenuItem("59.994 fps (2xNTSC)", new EventHandler(mnuFrameRates_Click));
    m.Checked = (frameRate == 59994);
    mnuFrameRates.MenuItems.Add(m);
    mnuFrameRates.Enabled = true;
}
catch { mnuFrameRates.Enabled = false; }
```

- **SubMenú Frame Size**

Añade al menú options submenú Video Capture Settings opción “Vídeo Frame Size”, el tamaño del marco utilizado para la captura de vídeo.

```
// Load frame sizes
```

```
try
{
    mnuFrameSizes.MenuItems.Clear();
    Size frameSize = capture.FrameSize;
    m = new MenuItem("160 x 120", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(160, 120));
    mnuFrameSizes.MenuItems.Add(m);
    m = new MenuItem("320 x 240", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(320, 240));
    mnuFrameSizes.MenuItems.Add(m);
    // Added a Pal format ...
    m = new MenuItem("352 x 288", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(352, 288));
    mnuFrameSizes.MenuItems.Add(m);
    m = new MenuItem("640 x 480", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(640, 480));
    mnuFrameSizes.MenuItems.Add(m);
    // Added a Ntsc format ...
    m = new MenuItem("720 x 480", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(720, 480));
    mnuFrameSizes.MenuItems.Add(m);
    mnuFrameSizes.Enabled = true;
    // Added some Pal formats ...
    m = new MenuItem("720 x 576", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(720, 576));
    mnuFrameSizes.MenuItems.Add(m);
```

```
    m = new MenuItem("768 x 576", new EventHandler(mnuFrameSizes_Click));
    m.Checked = (frameSize == new Size(768, 576));
mnuFrameSizes.MenuItems.Add(m);
}
catch { mnuFrameSizes.Enabled = false; }
```

- **SubMenú Audio Channels**

Añade al menú options submenú “Audio Channels”, los tipos de salida del audio. (Mono, Estéreo)

```
// Load audio channels
try
{
    mnuAudioChannels.MenuItems.Clear();
    short audioChannels = capture.AudioChannels;
    m = new MenuItem("Mono", new EventHandler(mnuAudioChannels_Click));
    m.Checked = (audioChannels == 1);
    mnuAudioChannels.MenuItems.Add(m);
    m = new MenuItem("Stereo", new EventHandler(mnuAudioChannels_Click));
    m.Checked = (audioChannels == 2);
    mnuAudioChannels.MenuItems.Add(m);
    mnuAudioChannels.Enabled = true;
capture.AudioSources = null;
}
catch { mnuAudioChannels.Enabled = false; }
```

- **SubMenú Audio Sampling Rate**

Añade al menú options submenú “Audio Sampling Rate”, el número de muetras de audio que se pueden tomar por segundo. (44.1 KHz, etc)

```
// Load audio sampling rate
try
{
    mnuAudioSamplingRate.MenuItems.Clear();
    int samplingRate = capture.AudioSamplingRate;
    m = new MenuItem("8 kHz", new EventHandler(mnuAudioSamplingRate_Click));
    m.Checked = (samplingRate == 8000);
    mnuAudioSamplingRate.MenuItems.Add(m);
    m = new MenuItem("11,025 kHz", new EventHandler(mnuAudioSamplingRate_Click));
    m.Checked = (capture.AudioSamplingRate == 11025);
    mnuAudioSamplingRate.MenuItems.Add(m);
    m = new MenuItem("22,05 kHz", new EventHandler(mnuAudioSamplingRate_Click));
    m.Checked = (capture.AudioSamplingRate == 22050);
    mnuAudioSamplingRate.MenuItems.Add(m);
    m = new MenuItem("32 kHz", new EventHandler(mnuAudioSamplingRate_Click));
    m.Checked = (capture.AudioSamplingRate == 32000);
    mnuAudioSamplingRate.MenuItems.Add(m);
    m = new MenuItem("44,1 kHz", new EventHandler(mnuAudioSamplingRate_Click));
    m.Checked = (capture.AudioSamplingRate == 44100);
    mnuAudioSamplingRate.MenuItems.Add(m);
    m = new MenuItem("48 kHz", new EventHandler(mnuAudioSamplingRate_Click));
m.Checked = (capture.AudioSamplingRate == 48000);
```

```
    mnuAudioSamplingRate.MenuItems.Add(m);
    mnuAudioSamplingRate.Enabled = true;
}
catch { mnuAudioSamplingRate.Enabled = false; }
```

- **SubMenú Audio Sampling Size**

Añade al menú options submenú “Audio Sampling SIze”, el número de bits que se puede obtener por muestra. (8 bits, 16 bits)

```
// Load audio sample sizes
try
{
    mnuAudioSampleSizes.MenuItems.Clear();
    short sampleSize = capture.AudioSampleSize;
    m = new MenuItem("8 bit", new EventHandler(mnuAudioSampleSizes_Click));
    m.Checked = (sampleSize == 8);
    mnuAudioSampleSizes.MenuItems.Add(m);
    m = new MenuItem("16 bit", new EventHandler(mnuAudioSampleSizes_Click));
    m.Checked = (sampleSize == 16);
    mnuAudioSampleSizes.MenuItems.Add(m);
    mnuAudioSampleSizes.Enabled = true;
}
catch { mnuAudioSampleSizes.Enabled = false; }
```

- **SubMenú Property Pages**

Añade al menú options submenú “Property Pages”, las propiedades del dispositivo de vídeo de entrada.

```
// Load property pages
try
{
    this.mnuPropertyPages.MenuItems.Clear();
    for (int c = 0; c < this.capture.PropertyPages.Count; c++)
    {
        p = this.capture.PropertyPages[c];
        m = new MenuItem(p.Name + "...", new EventHandler(this.mnuPropertyPages_Click));
        this.mnuPropertyPages.MenuItems.Add(m);
    }
    this.mnuPropertyPages.Enabled = (this.capture.PropertyPages.Count > 0);
}
catch { this.mnuPropertyPages.Enabled = false; }
```

- **SubMenú TV Tuner Channels**

Añade al menú options submenú “TV Tuner Channels”, muestra los canales de televisión almacenados en la base de datos.

```
// Load TV Tuner channels
try
{
```

```
    mnuChannel.MenuItems.Clear();
    if (this.TunerModeType == AMTunerModeType.TV)
    {
        int current_channel = this.tvSelections.CurrentChannel;
        for (int c = 1; c <= this.tvSelections.NbrTunerChannels; c++)
        {
            this.tvSelections.CurrentChannel = c;
            m = new MenuItem(this.tvSelections.GetChannelName.ToString(), new
                                    EventHandler(mnuChannel_Click));
            m.Checked = ((current_channel == c) && (this.capture.Tuner.Channel ==
                                    this.tvSelections.GetChannelNumber));
            mnuChannel.MenuItems.Add(m);
        }
        this.tvSelections.CurrentChannel = current_channel;
        mnuChannel.Enabled = true;
    }
    else
    {
        mnuChannel.Enabled = false;
}
}
catch { mnuChannel.Enabled = false; }
```

- **SubMenú TV Tuner Modes**

Añade al menú options submenú “TV Tuner Modes”, los modos de captura que puede tener el dispositivo. (TV o Radio Fm)

```
// Load Tuner Modes (such as TV and FM Radio
try
{
    this.menuTunerModes1.MenuItems.Clear();
    if ((this.capture.Tuner.AvailableAudioModes.TV) &&
                (this.capture.Tuner.AvailableAudioModes.FMRadio))
    {
        m = new MenuItem(AMTunerModeType.TV.ToString(), new
                        EventHandler(menuTunerModes1_Click));
        m.Checked = (this.TunerModeType == AMTunerModeType.TV);
        this.menuTunerModes1.MenuItems.Add(m);
        m = new MenuItem(AMTunerModeType.FMRadio.ToString(), new
                        EventHandler(menuTunerModes1_Click));
        m.Checked = (this.TunerModeType == AMTunerModeType.FMRadio);
        m.Enabled = false;
        this.menuTunerModes1.MenuItems.Add(m);
        this.menuTunerModes1.Enabled = true;
        this.menuTunerModes1.Visible = true;
    }
    else
    {
        this.menuTunerModes1.Enabled = false;
```

```
        this.menuTunerModes1.Visible = false;
    }
}
catch { this.menuTunerModes1.Enabled = false; this.menuTunerModes1.Visible = false; }
```

- **SubMenú TV Tuner Input Types**

Añade al menú options submenú “TV Tuner Input Types”, los modos de recepción de la señal. (Antena o Cable)

```
// Load TV Tuner input types
try
{
    mnuInputType.MenuItems.Clear();
    m = new MenuItem(TunerInputType.Cable.ToString(), new
                                    EventHandler(mnuInputType_Click));
    m.Checked = (capture.Tuner.InputType == TunerInputType.Cable);
    mnuInputType.MenuItems.Add(m);
    m = new MenuItem(TunerInputType.Antenna.ToString(), new
                                    EventHandler(mnuInputType_Click));
    m.Checked = (capture.Tuner.InputType == TunerInputType.Antenna);
    mnuInputType.MenuItems.Add(m);
    mnuInputType.Enabled = true;
}
catch { mnuInputType.Enabled = false; }
```

→ **Menú File**

- **SubMenú File Recording Mode**

Añade al menú file submenú “File Recording Mode Type”, los formatos de captura. (avi, wmv, wma)

```
// Load audio/video recording file modes
try
{
    menuAVRecFileModes.MenuItems.Clear();
    // Fill in all file modes, use enumerations also as string (and file extension)
    for (int i = 0; i < 3; i++)
    {
        m = new MenuItem(((DirectX.Capture.Capture.RecFileModeType)i).ToString(), new
                                        EventHandler(menuAVRecFileModes_Click));
        m.Checked = (i == (int)capture.RecFileMode);
        menuAVRecFileModes.MenuItems.Add(m);
    }
    menuAVRecFileModes.Enabled = true;
}
catch { menuAVRecFileModes.Enabled = false; }
```

Función Capture

```
//Capture

public Capture(Filter videoDevice, Filter audioDevice, bool audioViaPci)

{

if ( videoDevice == null && audioDevice == null )

throw new ArgumentException( "The videoDevice and/or the audioDevice parameter

                        must be set to a valid Filter.\n" );

this.videoDevice = videoDevice;

this.audioDevice = audioDevice;

this.Filename = getTempFilename();

this.AudioViaPci = audioViaPci;

createGraph();
}
```

Crea un nuevo objeto Capture. Pueden ser nulos cualquiera de los parámetros de audioDevice o videoDevice pero no ambos. Se puede habilitar el audio o no de la tarjeta de captura. La captura se guarda en un archivo temporal hasta que se indique el lugar donde se puede guardar.

Desde la aplicación se llama a esta función una vez seleccionado el dispositivo de captura del menú Resources, a través del siguiente código:

```
//Capture

Filter resources = null;
Filter audioDevice = null;
```

```
resources = capture.VideoDevice;
audioDevice = capture.AudioDevice;

if ((resources != null) || (audioDevice != null))

{

capture = new Capture(resources, audioDevice, AudioViaPci);

    capture.CaptureComplete += new EventHandler(OnCaptureComplete);

    capture.Filename = this.txtFilename.Text;

this.initMenu();
}
```

Una vez seleccionado el dispositivo en el menu resources se almacena en el objeto resource y se envía a la función capture. Se guarda la captura en un archivo en la ubicación que marca la opción filename.

5.2.2.4 Cambio de canales

El cambio de canales se los realiza pulsando los botones de los números. Los números marcados se muestran en el textbox inferior. Existe un retardo de 2 segundos para que se realice el cambio de canal, para poder marcar los canales de dos o tres dígitos si ese es el caso.

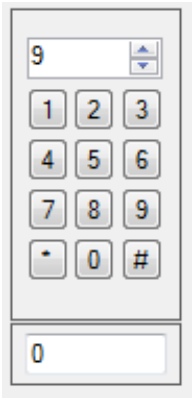


Figura V. 21 Control para cambio de canales

→ **Marcación del número uno**

El código es similar para los demás botones

```
//Marcación Button 1
if ((this.capture != null) && (this.capture.Tuner != null))
{
if (int.Parse(textBox1.Text) < 13)
    {
//channelUpDown.Value = channelUpDown.Value * 10 + 1;
        textBox1.Text = (int.Parse(textBox1.Text) * 10 + 1).ToString();
}
    else
    {
if (int.Parse(textBox1.Text) < 126 && int.Parse(textBox1.Text) > 12)
        {
textBox1.Text = (int.Parse(textBox1.Text) - int.Parse(textBox1.Text) + 1).ToString();
}
}
if (this.tunerInputType == TunerInputType.Antenna)
    {
if (int.Parse(textBox1.Text) > 1 && int.Parse(textBox1.Text) < 70)
       {
timer_channel.Interval = 2000;
           timer_channel.Start();
}
       else {  timer_channel.Stop();  }
}
```

```
   if (this.tunerInputType == TunerInputType.Cable)
   {
if (int.Parse(textBox1.Text) > 1 && int.Parse(textBox1.Text) < 126)
       {
timer_channel.Interval = 2000;
            timer_channel.Start();
}
}
}
```

→ **Control del tiempo del marcado**

```
//time_channel
timer_channel.Stop();
channelUpDown.Value = int.Parse(textBox1.Text);
textBox1.Text = "0";
```

5.2.2.5 Grabación Manual

Figura V. 22 Pantalla de añadir canales a la base de datos

→ **Botón Start**

Empieza la captura.

```
//Start
try
```

```
{
    string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
    //----------------------
    //-------txt_fileName--------
    SqlCeConnection con1 = new SqlCeConnection(conString);
    string sql1 = "SELECT location,filename FROM location";
    string loc = @"d:\", filename = "test";
    SqlCeCommand com1 = new SqlCeCommand(sql1, con1);
    con1.Open();
    SqlCeDataReader reader1 = com1.ExecuteReader();
    while (reader1.Read())
    {
        loc = (reader1.GetString(0));
        filename = (reader1.GetString(1));
    }
    con1.Close();
    txtFilename.Text = @loc + @filename;
    //-------file_recording_mode--------
    string sql2 = "SELECT recording_mode,video_compressor FROM file_recording_mode";
    string recor = @"Wmv", vid_com = "None";
    SqlCeCommand com2 = new SqlCeCommand(sql2, con1);
    con1.Open();
    SqlCeDataReader reader2 = com2.ExecuteReader();
    while (reader2.Read())
    {
        recor = (reader2.GetString(0));
```

```
        vid_com = (reader2.GetString(1));
    }
    con1.Close();
    txtFilename.Text += "." + recor.ToLower();
    string text = channelUpDown.Value.ToString() + "."
                    + "-" + DateTime.Now.Year.ToString()
                    + "-" + DateTime.Now.Month.ToString()
                    + "-" + DateTime.Now.Day.ToString() ;
    if (capture == null)
        throw new ApplicationException("Please select a video and/or audio device.");
    if (!capture.Cued)
       capture.Filename = @loc + filename + "." + text + "." + recor.ToLower();
    capture.Start();
    btn_Start.Enabled = false;
    btn_Stop.Enabled = true;
}
catch (Exception ex){ MessageBox.Show(ex.Message + "\n\n" + ex.ToString()); }
```

5.2.2.6 Automatizar grabación

Para automatizar la grabación será necesario almacenar la información con respecto a fechas, el canal y el tiempo que se necesita para grabar. Para este propósito una buena opción es utilizar bases de datos.

Oracle, SQL Server, Mysql, DB2, son claros ejemplos de bases de datos, pero para su utilización sería necesaria su instalación previa.

Una buena opción a elegir es Microsoft SQL Server Compact, desarrollado para usos en aplicaciones pequeñas.

Microsoft SQL Server Compact (SQL Server CE): Es un motor de base de datos relacional, de libre descarga y distribución, tanto para dispositivos móviles como para aplicaciones de escritorio. Especialmente orientada a sistemas ocasionalmente conectados, ofrece unas características útiles para aplicaciones ligeras. La versión más reciente es SQL Server Compact 3.5 SP2. Conocida como SQL Server CE o SQL Server Mobile. Desde la versión 2.0, el lanzamiento de SQL Server Compact ha ido ligado al de Microsoft Visual Studio .NET.

Una base de datos SQL Server Compact, a diferencia de una base de datos SQL Server se expone como un servicio de Windows, se ejecuta bajo el proceso de la aplicación que la utiliza (in-process). El tamaño máximo del archivo de base de datos es de 4 Gb y la extensión por defecto es .sdf. El nombre de la base de datos está limitado a 128 caracteres. En cuanto a limitaciones a destacar es su número máximo de tablas por base de datos que es de 1024, con un tamaño máximo por registro de 8060 bytes.

SQL Server Compact puede ser utilizada tanto por aplicaciones .NET Framework y .NET Compact Framework como por aplicaciones nativas. El conjunto de clases que proporciona es muy extenso y abarca desde la creación de la base de datos y mantenimiento hasta la manipulación de datos.

A través de la plataforma Visual Studio .NET se puede acceder utilizando el espacio de nombres System.Data.SqlServerCe.

El espacio de nombres System.Data.SqlServerCe es el proveedor de datos de .NET Compact Framework para SQL Server Mobile, que es una colección de clases que se utiliza en el entorno de desarrollo, administrado para proporcionar acceso a las bases de datos de SQL Server Mobile. Con System.Data.SqlServerCe, se pueden crear y administrar bases de datos de SQL Server Mobile en un dispositivo inteligente, así como establecer conexiones a las bases de datos de SQL Server.

La base de datos utilizada para la aplicación es TVisor.sdf. Con las siguientes tablas creadas:

- → **input_source (id, name)**

  Tabla que almacena los recursos de las tarjetas de video conectados. (S-Video TV, CATV).

- → **cannel_tv (id, number, name)**

  Tabla que almacena los números de canales y sus nombres. Canales de TV del 0 al 69.

- → **cannel_catv (id, number, name)**

  Tabla que almacena los números de canales, y sus nombres. Canales de CATV del 0 al 125.

- → **Schedule_record (id, input_source_id, channel, every, mode, year, month, day, hour, minute, time_hour, time_minute, time_sec)**

  Tabla que almacena los datos de los parámetros que se toman en cuenta para la grabación de los canales. Fecha, Hora, y el Tiempo de grabación además del canal y el recurso de los cuales se procede a grabar (tv o catv).

→ **sampling (id, channel_name, channel_number, input_source_id)**

→ **sampling_time (id, time_hour, time_minute, time_sec)**

Ambas tablas almacenan los datos para grabar un muestreo (sampling) de los canales escogidos en un tiempo determinado.

5.2.2.6.1 Ingreso de canales a la base de datos

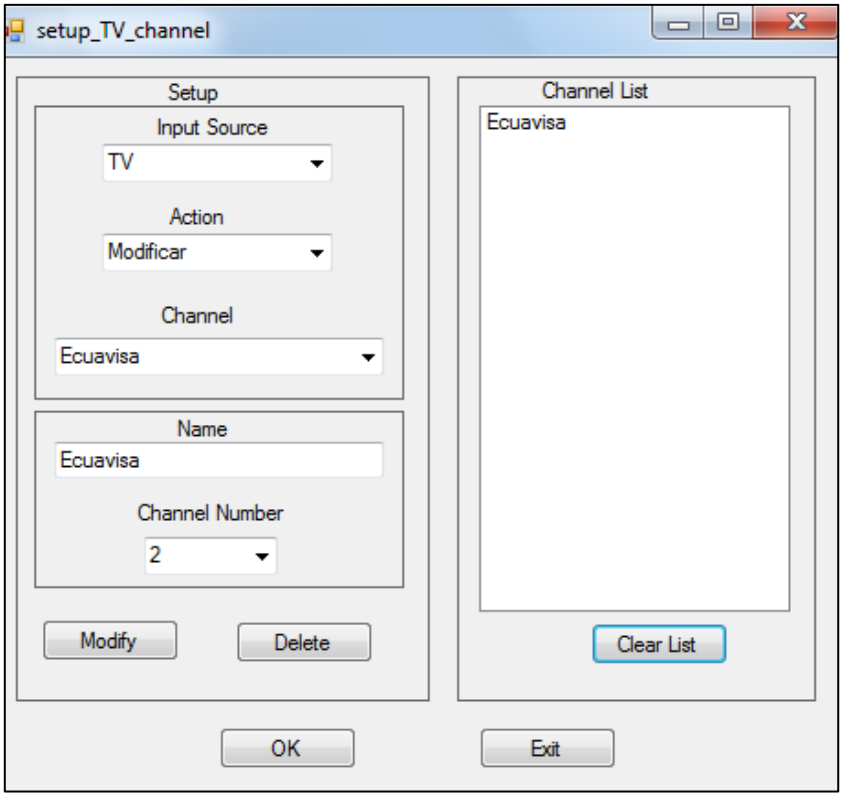


Figura V. 23 Pantalla de añadir canales a la base de datos

→ **Ingreso a la base de datos de los canales de TV y CATV**

```
// Añadir Nuevo canal
//consulta de datos a bd
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
using (SqlCeConnection conn = new SqlCeConnection(conString))
{
```

```
int id = 1;
string sql1="",sql2="";
if ((string)comboBox_input.Text == "TV")
{
    sql1 = "SELECT max(id) FROM channel_tv";
    //consultar valores duplicados
    sql2 = "SELECT name,number FROM channel_tv";
}
if ((string)comboBox_input.Text == "CATV")
{
    sql1 = "SELECT max(id) FROM channel_catv";
    //consultar valores duplicados
    sql2 = "SELECT name,number FROM channel_catv";
}
conn.Open();
SqlCeCommand com1 = new SqlCeCommand(sql1, conn);
SqlCeDataReader reader = com1.ExecuteReader();
while (reader.Read())
{
    if (!reader.IsDBNull(0)) { id = reader.GetInt32(0) + 1; }
    else { id = 1;  }
}
conn.Close();
conn.Open();
bool val = false,val1= false;
SqlCeCommand com2 = new SqlCeCommand(sql2, conn);
```

```
    SqlCeDataReader reader2 = com2.ExecuteReader();
    while (reader2.Read())
    {
        if (!reader2.IsDBNull(0))
        {
            if ((string)txt_name.Text == reader2.GetString(0)) { val=true; }
            if (int.Parse(comboBox_channel_number.Text) ==
                                                    reader2.GetSqlInt32(1))
           {
                val1=true;
            }
        }
    }
    conn.Close();
if (val)
   {
MessageBox.Show("El nombre del canal ya existe", "Channel
                        Name",MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
    else
    {
        if (val1)
        {
            MessageBox.Show("El número del canal ya existe", "Channel Number",
                                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
```

```
        else
        {
            //Insertar datos a bd
            string sql = "INSERT INTO channel_tv VALUES(@id,@number,@name)";
            string name = (string)txt_name.Text;
            conn.Open();
            SqlCeCommand com = new SqlCeCommand(sql, conn);
            com.Parameters.AddWithValue("@id", id);
            com.Parameters.AddWithValue("@number",
                                        comboBox_channel_number.Text);
            com.Parameters.AddWithValue("@name", name);
            com.ExecuteNonQuery();
            conn.Close();
            setup_TV_channel setup = new setup_TV_channel();
            setup.Show();
            this.Close();
        }
    }
}
```

→ **Eliminar canal de TV o CATV**

```
// Eliminar Nuevo canal
if (comboBox_channel.Text != null)
{
    int id = 1;
```

```
string sql1 = "", sql2 = "";
string name = (string)comboBox_channel.Text;
//Borrar datos a bd
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
SqlCeConnection conn = new SqlCeConnection(conString);
if ((string)comboBox_input.Text == "TV")
{
    sql1 = "SELECT id FROM channel_tv WHERE name = " +"'"+ name + "'";
    sql2 = "SELECT id FROM sampling WHERE channel_name = " + "'" + name
                                    + "'" + " AND input_source_id = 1";
}
if ((string)comboBox_input.Text == "CATV")
{
    sql1 = "SELECT id FROM channel_catv WHERE name = " + "'" + name + "'";
}
bool val = false;
conn.Open();
SqlCeCommand com1 = new SqlCeCommand(sql1, conn);
SqlCeDataReader reader = com1.ExecuteReader();
while (reader.Read())
{
    if (!reader.IsDBNull(0)) { id = reader.GetInt32(0); val = true; }
}
conn.Close();
if(val)
{
```

```
 // Open the connection using the connection string.
 using (SqlCeConnection con = new SqlCeConnection(conString))
{
    con.Open();
    if ((string)comboBox_input.Text == "TV")
    {
        sql1 = "DELETE FROM channel_tv WHERE id = " + id;
    }
    if ((string)comboBox_input.Text == "CATV")
    {
        sql1 = "DELETE FROM channel_catv WHERE id = " + id;
    }
    using (SqlCeCommand com = new SqlCeCommand(sql1, con))
    {
        com.ExecuteNonQuery();
        using (SqlCeCommand com2 = new SqlCeCommand(sql2, con))
        {
            //con.Open();
            string sql3 = "";
            int id_samplig = 0;
            SqlCeDataReader reader2 = com2.ExecuteReader();
            while (reader2.Read())
            {
                if (!reader2.IsDBNull(0))
                {
                    id_samplig = reader2.GetInt32(0);
```

```
                    sql3 = "DELETE FROM sampling WHERE id = " + id;
                    SqlCeCommand com3 = new SqlCeCommand(sql3, con);
                    com3.ExecuteNonQuery();
                }
            }
        }
        con.Close();
        setup_TV_channel form = new setup_TV_channel();
form.Show();
        this.Close();
}}}}
```

5.2.2.6.2 Resumen datos de grabación

Los datos de grabación se muestran realizando la consulta a la base de datos "schedule_record" y se los muestra en pantalla.

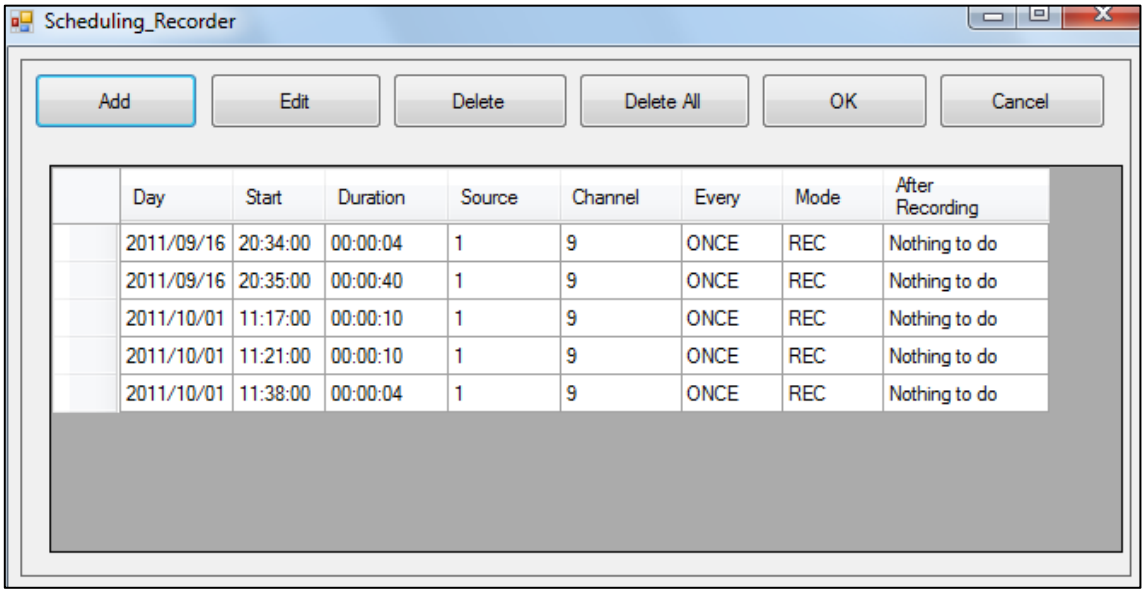


Figura V. 24 Pantalla de resumen datos de grabación

→ **Consulta a base de datos schedule_record**

```
// consulta a base de datos schedule_record
//Acceso a las propiedades de conexión a la base de datos
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
// Open the connection using the connection string.
SqlCeConnection con = new SqlCeConnection(conString);
int id=0, year=0, month=0, day=0, hour=0, minute=0, timer_hour=0,
                                    timer_minute=0, timer_sec, tv=0, catv=0, input=0;
con.Open();
string process = "", every = "", mode = "";
string sql = "SELECT year,month,day,hour,minute,time_hour,time_minute,
                                    time_sec,tv_id,catv_id,every,mode,process,id, input_source_id"
          +" FROM schedule_record" ;
SqlCeCommand com = new SqlCeCommand(sql, con);
SqlCeDataReader reader = com.ExecuteReader();
//-------------------------
DataTable Tabla = new DataTable(); //Se declara una variable de tipo DataTable
DataRow Renglon;//Esta variable de tipo DataRow
//Se agrega columnas a la variable Tabla que es de tipo DataTable
Tabla.Columns.Add(new DataColumn("ID"));
Tabla.Columns.Add(new DataColumn("Day"));
Tabla.Columns.Add(new DataColumn("Start"));
Tabla.Columns.Add(new DataColumn("Duration"));
Tabla.Columns.Add(new DataColumn("Source"));
Tabla.Columns.Add(new DataColumn("Channel"));
Tabla.Columns.Add(new DataColumn("Every"));
```

```
Tabla.Columns.Add(new DataColumn("Mode"));
Tabla.Columns.Add(new DataColumn("After Recording"));
while (reader.Read())
{
    year = (reader.GetInt32(0));
    month = (reader.GetInt32(1));
    day = (reader.GetInt32(2));
    hour = (reader.GetInt32(3));
    minute = (reader.GetInt32(4));
    timer_hour = (reader.GetInt32(5));
    timer_minute = (reader.GetInt32(6));
    timer_sec = (reader.GetInt32(7));
    tv=(reader.GetInt32(8));
    catv=(reader.GetInt32(9));
    every=(reader.GetString(10));
    mode=(reader.GetString(11));
    process=(reader.GetString(12));
    id = (reader.GetInt32(13));
    input = (reader.GetInt32(14));
string x="", y="", a="", b="", c="",d="",ee="";
//---------
    if(month < 10){ x = "0"; }
    if (day < 10){ y = "0"; }
    if (hour < 10){ a = "0"; }
    if (minute < 10){ b = "0"; }
    if (timer_hour < 10){ c = "0"; }
```

```
    if (timer_minute < 10){ d = "0"; }
if (timer_sec < 10){ ee = "0"; }
    //-----------------
    //Aqui se hace uso de la variable renglon, se inicializa diciendole
     que va a ser un nuevo renglon de la Tabla
    Renglon = Tabla.NewRow();
    //Aqui simplemente se agrega el renglon nuevo con los valores requeridos
Renglon[0] = id;
    Renglon[1] = year+"/"+ x +month+"/"+ y + day;
    Renglon[2] = a + hour + ":" + b + minute + ":" + "00";
    Renglon[3] = c + timer_hour + ":" + d + timer_minute + ":" + ee + timer_sec;
    Renglon[4] = input;
    if (tv != 0) { Renglon[5] = tv;}
    else { Renglon[5] = catv;}
    Renglon[6] = every;
Renglon[7] = mode;
    Renglon[8] = process;
    //Se agrega el renglon nuevo a la tabla
    Tabla.Rows.Add(Renglon);
}
//Aqui se le dice al dataGridView que tome la tabla y la muestre
dataGridView2.DataSource = Tabla;
dataGridView2.Columns[0].Visible = false;
con.Close();
```

5.2.2.6.3 Añadir un nuevo registro para la programación de grabación

Los datos de grabación se muestran realizando la consulta a la base de datos "schedule_record" y se los muestra en pantalla.

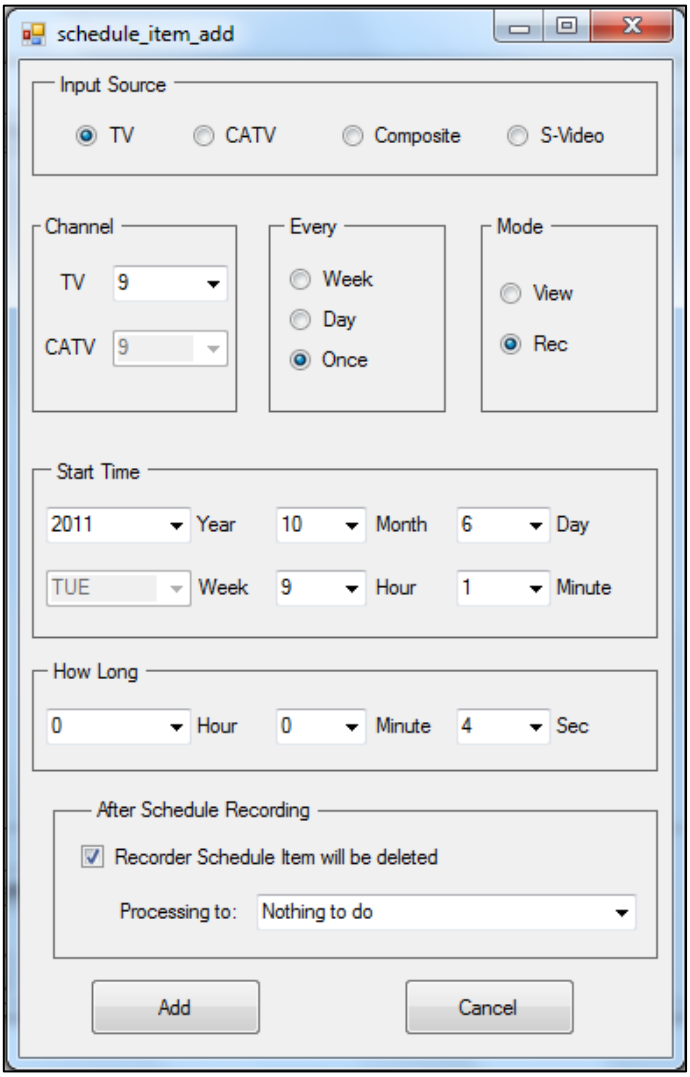


Figura V. 25 Pantalla de añadir nuevo registro para la programación de grabación

Al escoger el input source "TV" se habilita la selección TV para seleccionar el canal a grabar. Si es CATV se habilita la opción de CATV. Se puede escoger la selección de una vez, cada día o cada semana. A continuación se selecciona la fecha, hora y el tiempo de grabación.

→ **Añadir los parámetros a la base de datos**

```
//Añadir los parámetros a la base de datos
//Parámetros por defecto
int input = 1;
int tv = 9;
int catv = 0;
string every = "O";
string mode = "R";
//input source
if (rb_tv.Checked == true)
{
    input = 1;
    tv = int.Parse(comboBox_tv.Text.ToString());
    catv = 0;
    string week_day = "0";
}
else
{
    if (rb_catv.Checked == true)
    {
        input = 2;
        catv = int.Parse(comboBox_catv.Text.ToString());
        tv = 0;
        string week_day = "0";
    }
    else
```

```
    {
        if (rb_composite.Checked == true) { input = 3;}
        else { if (rb_svideo.Checked == true) {input = 4; } }
    }
}
//every
if (rb_week.Checked == true) { every = "WEEK"; }
if (rb_day.Checked == true) { every = "DAY"; }
if (rb_once.Checked == true) { every = "ONCE"; }
//mode
if (rb_view.Checked == true) { mode = "VIEW"; }
if (rb_rec.Checked == true) { mode = "REC"; }
//startime
int year = int.Parse(comboBox_year.Text.ToString());
int month = int.Parse(comboBox_month.Text.ToString());
int day = int.Parse(comboBox_day.Text.ToString());
int hour = int.Parse(comboBox_hour.Text.ToString());
int minute = int.Parse(comboBox_minute.Text.ToString());
int sec = 0;
int time_hour = int.Parse(comboBox_time_hour.Text.ToString());
int time_minute = int.Parse(comboBox_time_minute.Text.ToString());
int time_sec = int.Parse(comboBox_time_sec.Text.ToString());
string week = comboBox_week_day.Text.ToString();
string process = comboBox_process.Text.ToString();
//consulta de datos a bd
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
```

```
using (SqlCeConnection conn = new SqlCeConnection(conString))
{
    conn.Open();
    int id = 1;
    string sql1 = "SELECT max(id) FROM schedule_record";
    using (SqlCeCommand com1 = new SqlCeCommand(sql1, conn))
    {
        SqlCeDataReader reader = com1.ExecuteReader();
        while (reader.Read())
        {
            if (!reader.IsDBNull(0)) { id = reader.GetInt32(0) + 1; }
            else { id = 1; }
        }
        string sql2 = "DELETE FROM schedule_record_edit";
        using (SqlCeCommand com2 = new SqlCeCommand(sql2, conn))
        {
            com2.ExecuteNonQuery();
        }
        //Insertar datos a bd
        string sql = "INSERT INTO schedule_record
             VALUES(@id,@input_source_id,@tv_id,@catv_id,
                                                @every,@mode,@year,@month,@day"
                    + ",@hour,@minute,@sec,@time_hour,
                                                @time_minute,@time_sec,@week_day,@process)";

        using (SqlCeCommand com = new SqlCeCommand(sql, conn))
```

```
{
    com.Parameters.AddWithValue("@id", id);
    com.Parameters.AddWithValue("@input_source_id", input);
    com.Parameters.AddWithValue("@tv_id", tv);
    com.Parameters.AddWithValue("@catv_id", catv);
    com.Parameters.AddWithValue("@every", every);
    com.Parameters.AddWithValue("@mode", mode);
    com.Parameters.AddWithValue("@year", year);
    com.Parameters.AddWithValue("@month", month);
    com.Parameters.AddWithValue("@day", day);
    com.Parameters.AddWithValue("@hour", hour);
    com.Parameters.AddWithValue("@minute", minute);
    com.Parameters.AddWithValue("@sec", sec);
    com.Parameters.AddWithValue("@time_hour", time_hour);
    com.Parameters.AddWithValue("@time_minute", time_minute);
    com.Parameters.AddWithValue("@time_sec", time_sec);
    com.Parameters.AddWithValue("@week_day", week);
    com.Parameters.AddWithValue("@process", process);
    //com.Parameters.AddWithValue("@time",
                                        Convert.ToDateTime("7/4/1778"));
    com.ExecuteNonQuery();
    conn.Close();
    Scheduling_Recorder form = new Scheduling_Recorder();
    form.Show();
    this.Close();
}
```

```
}
conn.Close(); }
```

5.2.2.6.4 Realizar un muestreo de los canales

Se realiza una grabación de los canales especificados, en un tiempo determinado.

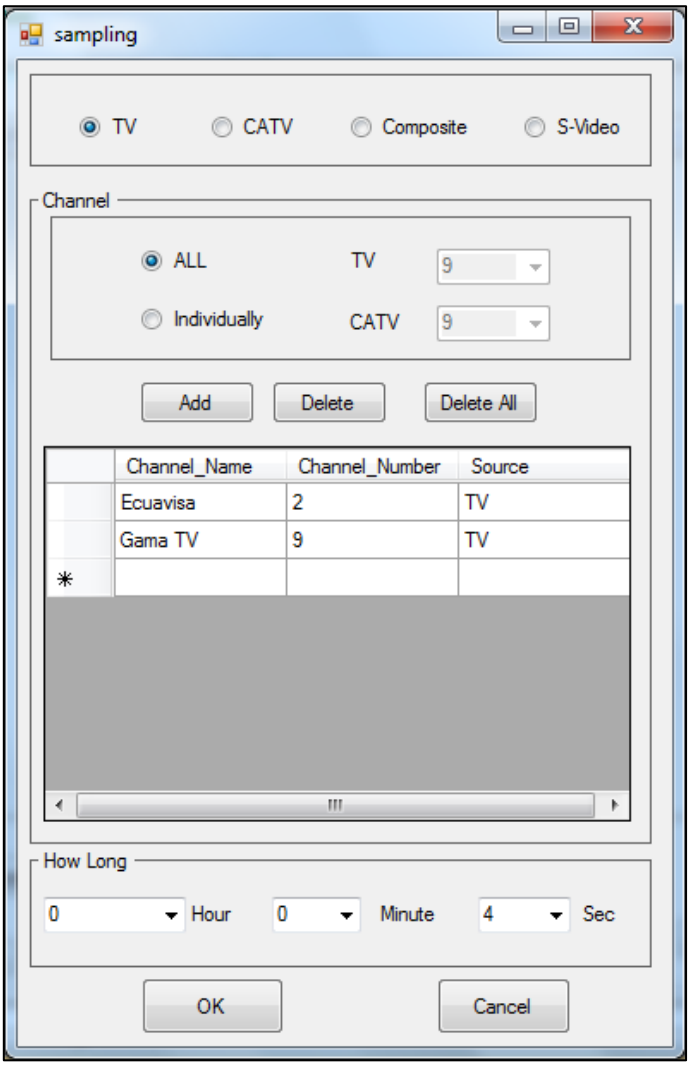


Figura V. 26 Pantalla de añadir canales para hacer un muestreo

→ Añadir los canales para el muestreo

```
//Añadir canales para el muestreo a la base de datos
if (rb_All.Checked == true)
```

```
{
string sql2 = "";
    int input = 1;
    if (rb_tv.Checked == true)
    {
input = 1;
        sql2 = "SELECT name, number FROM channel_tv";
}
    if (rb_catv.Checked == true)
    {
        input = 2;
        sql2 = "SELECT name, number FROM channel_catv";
    }
    //borrar de datos a bd
    string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
    SqlCeConnection conn = new SqlCeConnection(conString);
    conn.Open();
    string sql = "DELETE FROM sampling";
    SqlCeCommand com = new SqlCeCommand(sql, conn);
com.ExecuteNonQuery();
    conn.Close();
    //consulta de datos a bd
conn.Open();
    int id = 1, number = 1;
    string name = "";
    SqlCeCommand com1 = new SqlCeCommand(sql2, conn);
```

```
    SqlCeDataReader reader1 = com1.ExecuteReader();
    while (reader1.Read())
    {
 if (!reader1.IsDBNull(0))
{
 name = reader1.GetString(0);
              number = reader1.GetInt32(1);
              //Insertar datos a bd
              string sql1 = "INSERT INTO sampling
                            VALUES(@id,@channel_name,@channel_number,@input_source_id)";
SqlCeCommand com2 = new SqlCeCommand(sql1, conn);
                com2.Parameters.AddWithValue("@id", id);
                com2.Parameters.AddWithValue("@channel_name", name);
                com2.Parameters.AddWithValue("@channel_number", number);
                com2.Parameters.AddWithValue("@input_source_id", input);
                com2.ExecuteNonQuery();
}
            id = id + 1;
}
        sampling form = new sampling();
        form.Show();
        this.Close();
}
   else
   {
int input = 1;
```

```
        int channel = 9;
        //input source
        if (rb_tv.Checked == true)
        {
input = 1;
channel = int.Parse(comboBox_tv.Text.ToString());
}
        else
        {
if (rb_catv.Checked == true)
            {
input = 2;
                channel = int.Parse(comboBox_catv.Text.ToString());
}
}
        int time_hour = int.Parse(comboBox_time_hour.Text.ToString());
        int time_minute = int.Parse(comboBox_time_minute.Text.ToString());
        int time_sec = int.Parse(comboBox_time_sec.Text.ToString());
        //consulta de datos a bd
        string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
        SqlCeConnection conn = new SqlCeConnection(conString);
        conn.Open();
         int id = 1;
         string sql1 = "SELECT max(id) FROM sampling";
         SqlCeCommand com1 = new SqlCeCommand(sql1, conn);
         SqlCeDataReader reader = com1.ExecuteReader();
```

```
        while (reader.Read())
        {
            if (!reader.IsDBNull(0))  {  id = reader.GetInt32(0) + 1;   }
          else  {  id = 1; }
}
        conn.Close();
        conn.Open();
        string sql2 = "DELETE FROM schedule_record_edit";
        using (SqlCeCommand com2 = new SqlCeCommand(sql2, conn))
        {
com2.ExecuteNonQuery();
}
        conn.Close();
        //Insertar datos a bd
        string sql = "INSERT INTO sampling
                        VALUES(@id,@channel_name,@channel_number,@input_source_id)";
        string sql3 = "SELECT channel_number
                        FROM sampling WHERE input_source_id = " + input;
        string name = "No name";
        conn.Open();
        SqlCeCommand com3 = new SqlCeCommand(sql3, conn);
        SqlCeDataReader reader3 = com3.ExecuteReader();
        while (reader3.Read())
        {
if (!reader3.IsDBNull(0))
            {
```

```
int id_sampling = reader3.GetInt32(0);
                if (channel == id_sampling)
                {
MessageBox.Show("El canal ya está ingresado", "Channel Name",
                                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
                else
                {
string sql4 = "";
                    if (input == 1)
                    {
sql4 = "SELECT name FROM channel_tv WHERE number = " + channel;
}
                     if (input == 2)
                     {
sql4 = "SELECT name FROM channel_catv WHERE number = " + channel;
}
                      SqlCeCommand com4 = new SqlCeCommand(sql4, conn);
                      SqlCeDataReader reader4 = com4.ExecuteReader();
                      while (reader4.Read())
{
if (!reader4.IsDBNull(0))
                          {
    name = reader4.GetString(0);
break;
}
```

```
}}}}
        SqlCeCommand coma = new SqlCeCommand(sql, conn);
        coma.Parameters.AddWithValue("@id", id);
        coma.Parameters.AddWithValue("@channel_name", name);
        coma.Parameters.AddWithValue("@channel_number", channel);
        coma.Parameters.AddWithValue("@input_source_id", input);
        coma.ExecuteNonQuery();
        conn.Close();
        sampling form = new sampling();
form.Show();
        this.Close();
}
```

→ **Eliminar uno de los canales para el muestreo**

```
//Eliminar uno de los canales de la base de datos
if (dataGridView1.CurrentRow != null)
{
int id = int.Parse(dataGridView1.Rows
                        [dataGridView1.CurrentCell.RowIndex].Cells[0].Value.ToString());
//Insertar datoa a bd
    string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
    // Open the connection using the connection string.
    using (SqlCeConnection con = new SqlCeConnection(conString))
    {
con.Open();
```

```
        string sql = "DELETE FROM sampling WHERE id = " + id;
        using (SqlCeCommand com = new SqlCeCommand(sql, con))
        {
com.ExecuteNonQuery();
            con.Close();
            sampling form = new sampling();
form.Show();
            this.Close();
}}}
```

→ **Eliminar todos los canales para el muestreo**

```
//Eliminar todos canales de la base de datos
//Borrar datos a bd
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
// Open the connection using the connection string.
using (SqlCeConnection con = new SqlCeConnection(conString))
{
con.Open();
    //int id = 10;
    //string name = "Juan";
    string sql = "DELETE FROM sampling";
    string sql1 = "DELETE FROM schedule_record_edit";
    using (SqlCeCommand com = new SqlCeCommand(sql, con))
    {
com.ExecuteNonQuery();
```

```
        using (SqlCeCommand com1 = new SqlCeCommand(sql1, con))
        {
com1.ExecuteNonQuery();
}
        con.Close();
        sampling form = new sampling();
        form.Show();
        this.Close();
}}
```

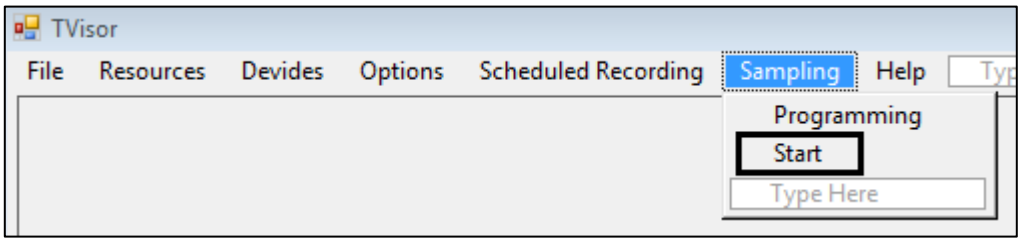


Figura V. 27 Botón empezar muestreo

→ Ejecutar el muestreo de los canales

```
//Ejecutar el muestreo
timer_load_samplig.Stop();
//timer_stop_sampling.Enabled = true;
try
{
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
    SqlCeConnection con = new SqlCeConnection(conString);
    int timer_hour = 0, timer_minute = 0, timer_sec = 0, tv = 0, input = 0;
    con.Open();
```

```
    //--------------sampling_time-----------
    string sql = "SELECT timer_hour,timer_minute,timer_sec FROM sampling_time";
    SqlCeCommand com = new SqlCeCommand(sql, con);
   SqlCeDataReader reader = com.ExecuteReader();
    while (reader.Read())
    {
timer_hour = (reader.GetInt32(0));
          timer_minute = (reader.GetInt32(1));
          timer_sec = (reader.GetInt32(2));
}
     con.Close();
     con.Open();
     //--------------sampling_time-----------
     string sql1 = "SELECT max(channel_number) FROM sampling" ;
     string name = "";
     SqlCeCommand com1 = new SqlCeCommand(sql1, con);
SqlCeDataReader reader1 = com1.ExecuteReader();
    while (reader1.Read())
    {
if (!reader1.IsDBNull(0))
        {
 tv = (reader1.GetInt32(0));
             string sqla = "SELECT input_source_id, channel_name
                      FROM sampling WHERE channel_number = " + tv  ;
              SqlCeCommand coma = new SqlCeCommand(sqla, con);
              SqlCeDataReader readera = coma.ExecuteReader();
```

```
            while (readera.Read())
            {
                input = (readera.GetInt32(0));
                 name = readera.GetString(1);
            }
            //-----------------------
            timer_stop_sampling.Interval =
                     (timer_hour * 3600 + timer_minute * 60 + timer_sec) * 1000;
            int channel = int.Parse(tv.ToString());
            //int channel = 1;
            if (input == 1)
            {
if (this.tunerInputType != TunerInputType.Antenna)
                {
this.tunerInputType = TunerInputType.Antenna;
                    capture.Tuner.InputType = this.tunerInputType;
}
                channel = int.Parse(tv.ToString());
}
            if (input == 2)
            {
    if (this.tunerInputType != TunerInputType.Cable)
                {
this.tunerInputType = TunerInputType.Cable;
                    capture.Tuner.InputType = TunerInputType.Cable;
                    this.DefaultChannel = channel;
```

```
                    channelUpDown.Value = channel;
}
                channel = int.Parse(tv.ToString());
}
            if ((this.capture != null) && (this.capture.Tuner != null))
            {
if (this.DefaultChannel != channel)
                {
this.DefaultChannel = channel;
this.capture.Tuner.Channel = this.DefaultChannel;
//this.capture.Tuner.TuningSpace = this.defaultTuningSpace;
                    channelUpDown.Value = channel;
}
                //this.updateMenu();
}
            //-----------------------
string text = "d:"+ "/"+ name +"_" +  channel.ToString()
                            + "-" + DateTime.Now.Year.ToString()
+ "-" + DateTime.Now.Month.ToString()
+ "-" + DateTime.Now.Day.ToString()
+ ".wmv";
txtFilename.Text = text;
            if (capture == null)
                throw new ApplicationException("Please select a video and/or audio
device.");
            if (!capture.Cued)
```

```
                capture.Filename = txtFilename.Text;
            capture.Start();
            btn_Cue.Enabled = false;
            btn_Start.Enabled = false;
            btn_Stop.Enabled = true;
            timer_stop_sampling.Start();
            //timer_load_samplig.Enabled = false;
}
}
    con.Close();
    this.updateMenu();
}
catch (Exception ex)
{
MessageBox.Show(ex.Message + "\n\n" + ex.ToString());
}
```

5.2.2.6.5 Opciones por defecto

→ **Selección del dispositivo por defecto**

Selección del dispositivo de entrada por defecto.

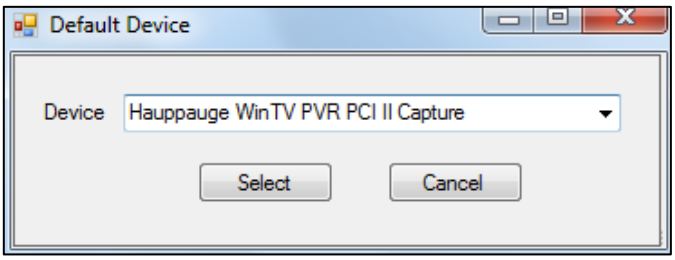


Figura V. 28 Selección del dispositivo por defecto

```
//Seleccionar el dispositivo por defecto
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
SqlCeConnection conn = new SqlCeConnection(conString);
string sql2 = "DELETE FROM default_device";
conn.Open();
SqlCeCommand com2 = new SqlCeCommand(sql2, conn);
com2.ExecuteNonQuery();
conn.Close();
string sql = "INSERT INTO default_device VALUES(@id,@name)";
conn.Open();
SqlCeCommand com = new SqlCeCommand(sql, conn);
com.Parameters.AddWithValue("@id", 1);
com.Parameters.AddWithValue("@name", cb_device.Text);
com.ExecuteNonQuery();
conn.Close();
this.Close();
```

→ **Selección de la localización de los archivos a guardar por defecto**

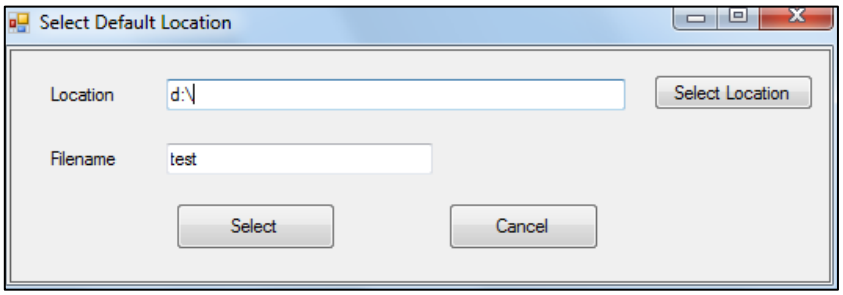


Figura V. 29 Selección de la localización de los archivos a guardar por defecto

```
//Selección de la localización de los archivos a guardar por defecto
SqlCeConnection con = new SqlCeConnection(conString);
string sql = "DELETE FROM location";
con.Open();
SqlCeCommand com = new SqlCeCommand(sql, con);
com.ExecuteNonQuery();
con.Close();
sql = "INSERT INTO location VALUES(@id,@location,@filename)";
con.Open();
SqlCeCommand com1 = new SqlCeCommand(sql, con);
com1.Parameters.AddWithValue("@id", 1);
com1.Parameters.AddWithValue("@location",txtLocation.Text);
com1.Parameters.AddWithValue("@filename",txtFilename.Text);
com1.ExecuteNonQuery();
con.Close();
this.Close();
```

→ **Selección del formato de grabación y el compresor por defecto**

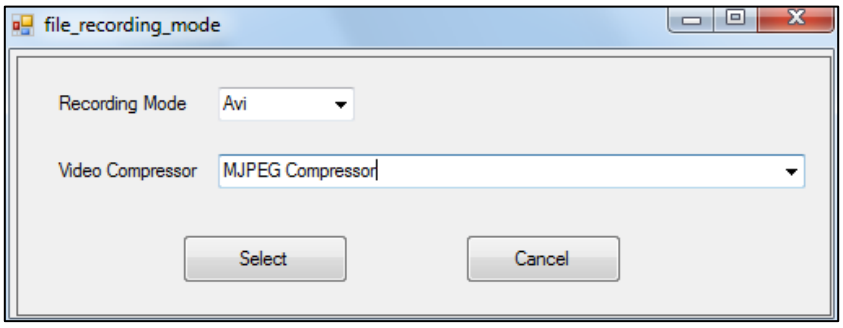


Figura V. 30 Selección del formato de grabación y el compresor por defecto

```
// Selección del formato de grabación y el compresor por defecto
SqlCeConnection con = new SqlCeConnection(conString);
string sql = "DELETE FROM file_recording_mode";
con.Open();
SqlCeCommand com = new SqlCeCommand(sql, con);
com.ExecuteNonQuery();
con.Close();
sql = "INSERT INTO file_recording_mode
                    VALUES(@id,@recording_mode,@video_compressor)";
con.Open();
SqlCeCommand com1 = new SqlCeCommand(sql, con);
com1.Parameters.AddWithValue("@id", 1);
com1.Parameters.AddWithValue("@recording_mode", cb_recording_mode.Text);
com1.Parameters.AddWithValue("@video_compressor", cb_video_compressor.Text);
com1.ExecuteNonQuery();
con.Close();
this.Close();
```

→ Selección de opciones de sintonización por defecto

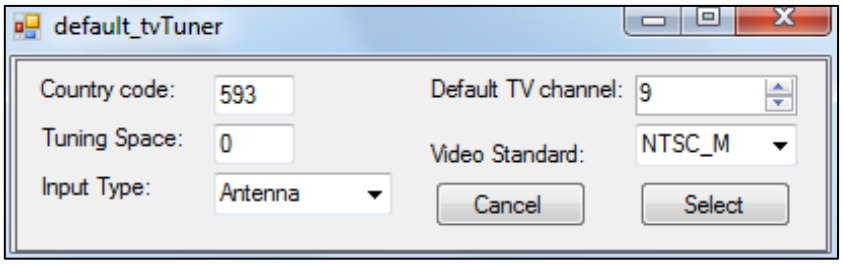


Figura V. 31 Selección de opciones de sintonización por defecto

```
// Selección del opciones de sintonización por defecto
string conString = Properties.Settings.Default.tvisor_databaseConnectionString;
SqlCeConnection conn = new SqlCeConnection(conString);
string sql2 = "DELETE FROM default_tuner";
conn.Open();
SqlCeCommand com2 = new SqlCeCommand(sql2, conn);
com2.ExecuteNonQuery();
conn.Close();
string sql = "INSERT INTO default_tuner
            VALUES(@id,@country_code,@tunning_space,@input_type,@default_channel,
                                    @video_standard)";
conn.Open();
SqlCeCommand com = new SqlCeCommand(sql, conn);
com.Parameters.AddWithValue("@id", 1);
com.Parameters.AddWithValue("@country_code", textCountryCode1.Text);
com.Parameters.AddWithValue("@tunning_space", textTuningSpace1.Text);
com.Parameters.AddWithValue("@input_type", listboxInputType1.Text);
com.Parameters.AddWithValue("@default_channel", numericUpDownA.Value);
com.Parameters.AddWithValue("@video_standard", listboxVideoStandard1.Text);
com.ExecuteNonQuery();
conn.Close();
this.Close();
}
```

CONCLUSIONES

1. Windows es el sistema operativo más utilizado entre los usuarios de PC, considerándose casi un estándar no oficial. Esta es la principal razón para el desarrollo de la aplicación sobre esta plataforma.

2. El software fue desarrollado utilizando la herramienta Visual Studio.NET, a través del lenguaje C#, aprovechando las funciones de DirectX, que ofrece acceso a tecnologías claves para simplificar el desarrollo de aplicaciones.

3. DirectX ofrece funcionalidades útiles multimedia que se pueden utilizar para el desarrollo de software que hacen uso de gráficos, audio y/o video. El uso de librerías DirectX proporciona acceso de bajo nivel al hardware multimedia de forma independiente del dispositivo. Esta ventaja permite dar soporte para trabajar con cualquier tipo de sintonizadora de TV.

4. La aplicación permite el monitoreo de los canales de televisión. Ofrece funciones de grabación en distintas modalidades con selección de fecha y hora además del tiempo de duración.

5. Para el almacenamiento de la información de grabación se utilizó base de datos SQL Server CE. SQL Server CE es un motor de base de datos relacional, de libre descarga y distribución, tanto para dispositivos móviles como para aplicaciones

de escritorio. Especialmente orientada a sistemas ocasionalmente conectados, ofrece unas características útiles para aplicaciones ligeras.

6. El sistema planteado permite el ahorro sustancial de recursos tanto humano como de infraestructura técnica para el monitoreo y control de medios televisivos.

7. Las estrategias básicas para reducir la cantidad de datos de un vídeo son: comprimir los datos, reducir las dimensiones de imagen del vídeo o reducir la velocidad de fotogramas del vídeo capturado.

8. La aplicación utiliza un método de comprensión predefinido en el formato wmv con características de velocidad de 700kbps, un ancho de imagen de 352 pixeles, altura de la imagen de 288 pixeles, velocidad de imágenes de 30 fps, con este conjunto de propiedades se obtiene una buena señal de video y obviamente contribuye en el ahorro de espacio en el disco duro.

RECOMENDACIONES

1. Se recomienda el uso aplicativo de este sistema en entidades del Estado como la SUPERTEL, empresas que anuncian sus productos en la televisión, agencias de publicidad que necesitan monitorear las emisiones que se transmiten a través de la TV y porque no al espectador común que por sus actividades diarias no acceden a toda la oferta televisiva que le gustaría.

2. Para el lector de esta memoria que se interese en seguir desarrollando el software se suguiere revisar la bibliografía referente a el uso de DirectX, ya que es el punto de partida del sistema. Asi como inspeccionar periódicamente la página Code Project en la que constantemente programadores de todo el mundo se encuentran aportando nuevas ideas o soluciones para el desarrollo de software.

3. Es muy útil para el usuario revisar previamente el manual de la aplicación para una correcta comprensión de todas las funcionalidades que dispone este sistema.

4. Si bien es posible conectar a la tarjeta sintonizadora de TV una antena de televisión común y corriente, se recomienda utilizar un dispositivo amplificador de la señal de entrada para obtener una máxima calidad y el menor ruido posible.

5. La aplicación ofrece funciones para modificar las características de video y audio. Es recomendable modificar todos estos parámetros antes de iniciar la captura.

6. Es recomendable el uso de tarjetas sintonizadoras de TV que dispongan de compresión interna de video (compresión por hardware). Esta característica reduce significativamente el uso de recursos del computador.

7. Se recomienda utilizar las herramientas de DirectX para el desarrollo futuro de aplicaciones para televisión digital.

RESUMEN

Se diseñó e implementó un prototipo de grabación automatizado de señal de televisión abierta. Este proyecto permitirá facilitar el trabajo de entidades que necesitan monitorear los medios televisivos.

Se aplicó el método inductivo - deductivo. Partiendo del problema de falta de personal y recursos tecnológicos para realizar el monitoreo de la programación de televisión, satisfaciendo los objetivos planteados.

La aplicación se desarrollo bajo la plataforma Visual Studio.NET, a través del lenguaje C#. Se basa en la utilización de librerías DirectX, que proporcionan acceso de bajo nivel al hardware multimedia de forma independiente del dispositivo.

Para el almacenamiento de la información de grabación se utilizó base de datos SQL Server CE.

El software implementado ofrece funciones para la grabación programable. Permite la selección de fechas y horas además del tiempo de duración para monitorear y/o grabar la señal de televisión abierta de manera rápida y sencilla.

Como resultado, el sistema permite el ahorro sustancial de recursos tanto humano como de infraestructura técnica.

Este sistema puede ser utilizado por entidades del Estado como la SUPERTEL. Así como por agencias de publicidad que necesitan monitorear las emisiones que se transmiten a través de la TV. Además, las personas que por sus actividades diarias no acceden a toda la oferta televisiva que le gustaría.

Se recomienda a la persona interesada en seguir desarrollando el software revisar la bibliografía referente al uso de DirectX.

SUMMARY

This system allows significant savings in human and infrastructure techniques resources and eases the work of entities that need to monitor the television media. However, the problem we face is the lack of resources to carry out a monitoring of the Television programming system.

Our objective is to find the right equipment and tools that will allow the implementation of this prototype in order to serve different entities. In order to find favorable results, we applied an inductive-deductive method.

As a result, the system we propose allows a significant saving of technological infrastructure and human resources.

We recommend private and state entities to use this proposed system if their need is to monitor emissions that are transmitted through the television.

GLOSARIO

ASF

Advanced Systems Format es el formato propiedad de Microsoft para contener audio y video.

AVI

Es un formato de Windows desarrollado por Microsoft, que puede almacenar simultáneamente un flujo de datos de video y varios flujos de audio.

Cinepak

Su fabricante es Radius Corp. aunque es un codec algo antiguo todavía proporciona video de buena calidad a pantalla completa alta velocidad de muestreo y profundidad de muestra.

Codificación

La codificación consiste en la traducción de los valores de tensión eléctrica analógicos que ya han sido cuantificados o ponderados al sistema binario, mediante códigos preestablecidos. La señal analógica va a quedar transformada en un tren o cadena de impulsos digitales (sucesión de ceros y unos).

Codificación Inter-Frame

O compresión temporal, aprovecha la ventaja que existe cuando las imágenes sucesivas son similares. En lugar de enviar la información de cada imagen por

separado, el codificador Inter envía la diferencia existente entre la imagen previa y la actual en forma de codificación diferencial.

COM

Component Object Mode es un estándar creado por Microsoft que define a nivel binario como los objetos se deben crear, destruir e interactuar entre ellos.

Componente

Es la implementación de una o más interfaces y queda definida por las interfaces que implementa. La herencia de un componente se concibe a nivel de interfaces.

Compresión Intra-Frame

O compresión espacial, comprime cada fotograma por separado.

Cuantificación

La cuantificación lo que hace es convertir una sucesión de muestras de amplitud continua en una sucesión de valores de amplitudes discretas, o lo que es lo mismo, en una señal digital, aunque no binaria.

Delta Frames

Conienen trozos o partes de la información que se forman a partir de los keyframes.

Direct3D®

Proporciona interfaces de alto y bajo nivel para generar polígonos con texturas en 3D por software y por hardware.

DirectDraw®

Proporciona animación realista usando intercambio de páginas de video, acceso a coprocesadores gráficos especializados y administración de la memoria de video. También sirve de base para otros componentes como DirectShow® y Direct3D®.

DirectPlay®

Incluye servicios transparentes de mensajería independientes del medio para crear juegos con varios jugadores, así como las funciones necesarias para organizar y ejecutar un juego multijugador.

DirectShow®

Proporciona una interfaz para el manejo de flujos multimedia provenientes de archivos o dispositivos de adquisición de audio y video.

DirectShowNET

El objetivo de esta biblioteca es permitir el acceso a las funcionalidades de Microsoft DirectShow desde .NET.

DirectSound®

Proporciona sonido estéreo y 3D con mezcla de sonido por hardware, así como administración de la memoria de la tarjeta de sonido.

DirectX®

Desarrollado por Microsoft® es el software que permite manipular el hardware del sistema a bajo nivel.

DirectX.Capture

Es una biblioteca de clases para la captura de audio y vídeo a los archivos AVI en formato .NET.

DLL

Es el término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación es exclusiva a los sistemas operativos Windows siendo ".dll" la extensión con la que se identifican estos ficheros.

Filter

Es un objeto COM y se ubica en un arreglo denominado gráfico de filtros (filter graph) el cual proporciona además una idea global del funcionamiento e interacción entre filtros.

Filter Graph

Un conjunto de filtros conectados se denomina gráfico de filtros (filter graph). La aplicación no tiene que manejar los filtros individuales en el gráfico de filtros. En lugar de eso, DirectShow provee de un componente llamado Manejador del Gráfico de Filtros (Filter Graph Manager).

Frame

Se denomina Frame a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación. La continua sucesión de estos fotogramas producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos.

GOP

Group Of Pictures o Grupo de Imágenes, es una secuencia de imágenes que está constituida por una imagen inicial (key frame) "I" y las siguientes imágenes "P" (delta frames) hasta el comienzo de otra imagen "I"

GUID

Es un número pseudoaleatorio empleado en aplicaciones de software. Aunque no se puede garantizar que cada GUID generado sea único, el número total de claves únicas (2^{128} o $3.4028\ x\ 10^{38}$) es tan grande que la posibilidad de que se genere un mismo número dos veces puede considerarse nula en la práctica.

Interface

Es un conjunto de métodos (funciones) y atributos (datos) que tienen una fuerte relación lógica entre ellos.

Key Frame

Un keyframe es una imagen completa con todos sus atributos.

Ligos Indeo

Códec de vídeo desarrollado inicialmente por Intel, pero adquirido posteriormente por Ligos

MP3

Permite hacer una pista de audio en formato Compact Disk, ocupando muy poco espacio y sin apenas pérdida de calidad

MPEG

Motion Picture Experts Group, en español Grupo de Expertos de Imágenes en Movimiento, es un grupo de estándares empleados para la codificación de información audiovisual

Muestreo

El muestreo digital es uno de los procesos que permite la digitalización de las señales.

Parseo

El parseo transforma una entrada de texto en una estructura de datos (usualmente un árbol) que es apropiada para ser procesada. Generalmente los parseadores primero identifican los símbolos de la entrada y luego construyen el árbol de parseo para esos símbolos.

Pins

Los pins son responsables de proveer interfaces para conectarse con otros pins y para transportar los datos.

Señal de crominancia

Es el componente de la señal de vídeo que contiene las informaciones del color. Comprende los canales rojo, verde y azul.

Señal de luminancia

Es la que recibe la información con toda la intensidad luminosa y se representa la imagen en blanco y negro con todos los tonos mediante la escala de grises. No contiene información sobre los colores de la imagen.

Subsampling

Denominado también sub-muestreo, consiste en reducir la información de color preservando intacta la luminosidad

Tarjeta sintonizadora o capturadora

Es un periférico que permite ver los distintos tipos de televisión o señales de video en la pantalla del computador, mediante programas instalados que ayudan al usuario a utilizar la tarjeta.

Video For Windows

Es un framework multimedia que añadió por primera vez a Microsoft Windows el soporte para operar con ficheros de video.

The Code Project

Es un portal formado por una comunidad enorme de miembros, en la que todo el mundo puede aportar sus conocimientos en diversas áreas de desarrollo: C++, C#, MFC, ASP, ASP .NET y la plataforma .NET en general.

YUV

La "Y" hace referencia a la luminosidad, mientras que "U" y "V" a la crominancia o color.

BIBLIOGRAFÍA

1. PESCE, Mark D., Programming Microsoft DirectShow for Digital Video and Television., Washington – USA., Microsoft Corporation., 2003., 448p.
2. SHELLY, Gary., Visual Basic 2008., Boston – USA., Course Technology, Cengage Learning., 2009., Pp 1 – 470.
3. TARRÉS, Francesc., Sistemas Audiovisuales - Televisión Analógica y Digital., Barcelona - España., Edición de la Universidad Politécnica de Cataluña., 2000., Pp 1 – 376.
4. WATSONET, Karli., Beginning Visual C# 2005., Indiana – USA., Wiley Publishing, Inc., 2005. Pp 1 – 575.
5. CASTRO, Manuel A., Vídeo Digital., Universidad Nacional de Educación a Distancia., España., 2008., 110p
6. ZAIDÁN, M., Análisis del dividendo digital resultante de la migración de la televisión analógica a digital en el Ecuador., Tesis Ing. Telecomunicaciones., Quito Escuela Politecnica Nacional., Escuela de Ingeniería Eléctrica y Electrónica., 2010. Pp 1 – 75
7. CORONILLA, U., Procesamiento digital de video en tiempo real y "video wall" con la PC., Tesis Maestro en Ciencias de la Computación., Mexico Universidad Autonoma Metropolitana AZCAPOTZALCO., 2005. Pp 29 - 46
8. TELEVISIÓN

   http://webs.uvigo.es/xuliofh/Web-Tv/TvIndex.html

(2011-07-01)

9. INFORMACIÓN BÁSICA DE TELEVISIÓN ABIERTA Y PAGADA

http://www.supertel.gob.ec/index.php?option=com_content&view=article&id=234:informacion-basica-television-pagada&catid=68:radiodifusion-sonora&Itemid=43

(2011-07-07)

10. INTERFACES DE DIRECTX

http://msdn.microsoft.com/en-us/library/dd743590%28v=VS.85%29.aspx

(2011-07-10)

11. CODECS Y FORMATOS

http://es.scribd.com/doc/54866744/26/CODECS-Y-FORMATOS

(2011-07-11)

12. TARJETAS SINTONIZADORAS DE TELEVISIÓN

http://es.wikipedia.org/wiki/Tarjeta_sintonizadora_de_televisi%C3%B3n

(2011-07-14)

13. TARJETAS SINTONIZADORAS DE TELEVISIÓN

http://www.tv-cards.com/messageboard/

(2011-07-20)

14. TARJETA SINTONIZADORA D-LINK

http://www.dlinkla.com/home/productos/producto.jsp?idp=467

(2011-07-28)

15. TARJETA SINTONIZADORA HAUPPAUGE PVR-150

http://www.mythtv.org/wiki/Hauppauge_PVR-150

(2011-08-05)

16. VIDEO FOR WINDOWS

http://es.wikipedia.org/wiki/Video_for_Windows

(2011-08-11)

17. SQL SERVER CE

http://es.wikipedia.org/wiki/SQL_Server_Compact

(2011-08-30)

18. SQL SERVER CE

http://msdn.microsoft.com/es-es/library/system.data.sqlserverce(v=vs.80).aspx

(2011-09-02)

19. VISUAL STUDIO .NET - CONTROL LIST BOX

http://www.elguille.info/colabora/NET2005/Perxi_Usando_el_control_ListBox.htm

(2011-09-11)

20. THE CODE PROJECT - SOFTWARE DEVELOPT

http://www.codeproject.com/

(2011-09-15)

21. CREAR TAREAS PROGRAMADAS DESDE C#

http://escarbandocodigo.wordpress.com/2009/10/21/crear-tareas-programadas-desde-c/

(2011-09-17)

ANEXOS

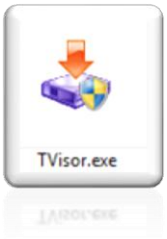
MANUAL DE USUARIO

TVisor es una aplicación que ofrece una solución completa para sintonizar TV analógica en tu PC.

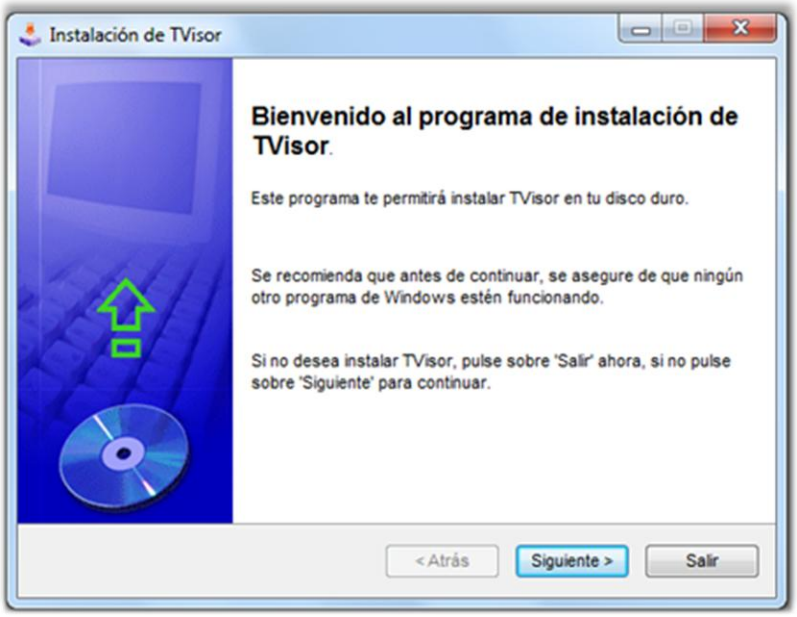
Antes de la instalación de este software, es necesario instalar los drivers de la tarjeta sintonizadora de TV que se disponga, para que funcione correctamente.

⇒ **Instalando el Software**

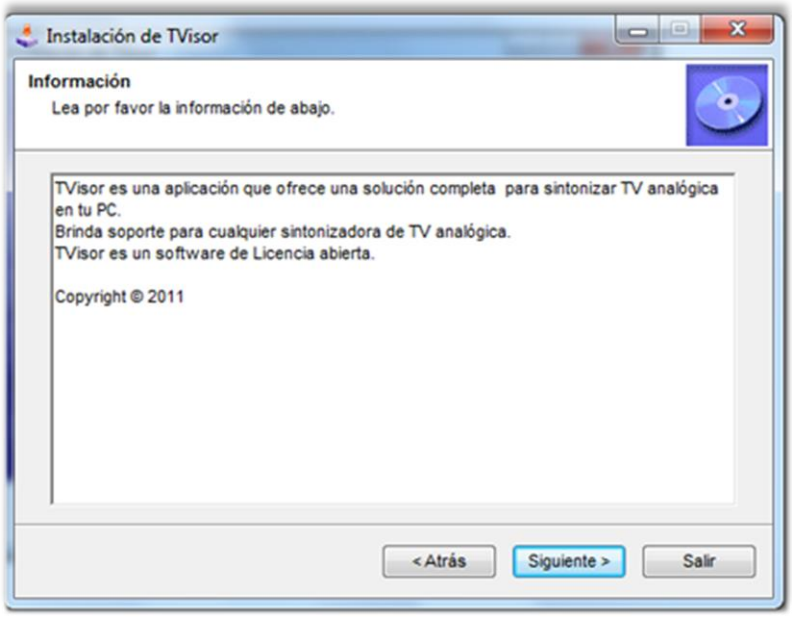
Paso 1: Ejecutar el instalador



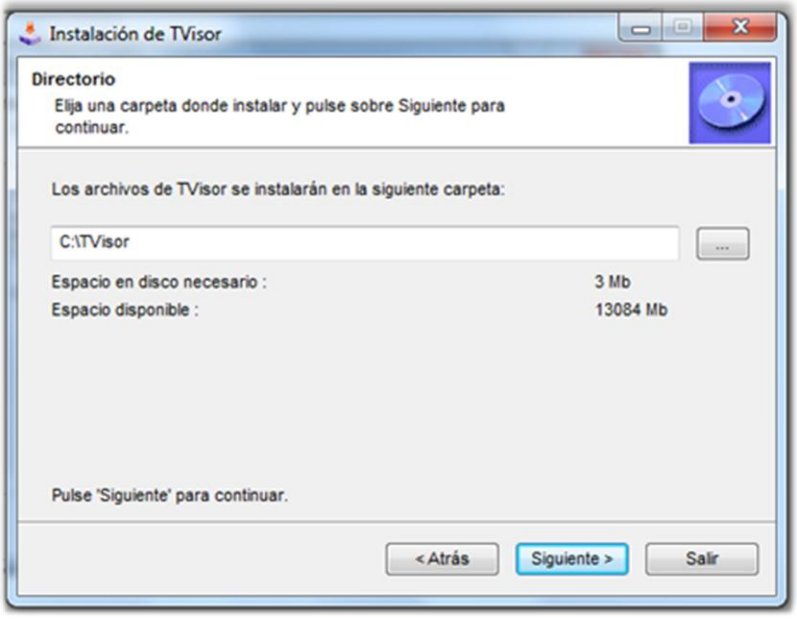
Paso 2: Click “Siguiente” para continuar con la instalación.



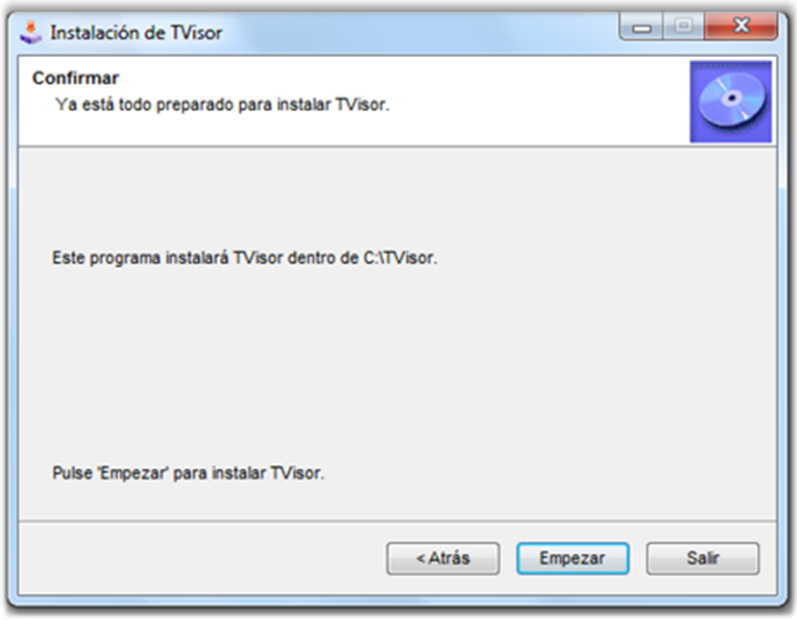
Paso 3: Click “Siguiente” para continuar con la instalación.



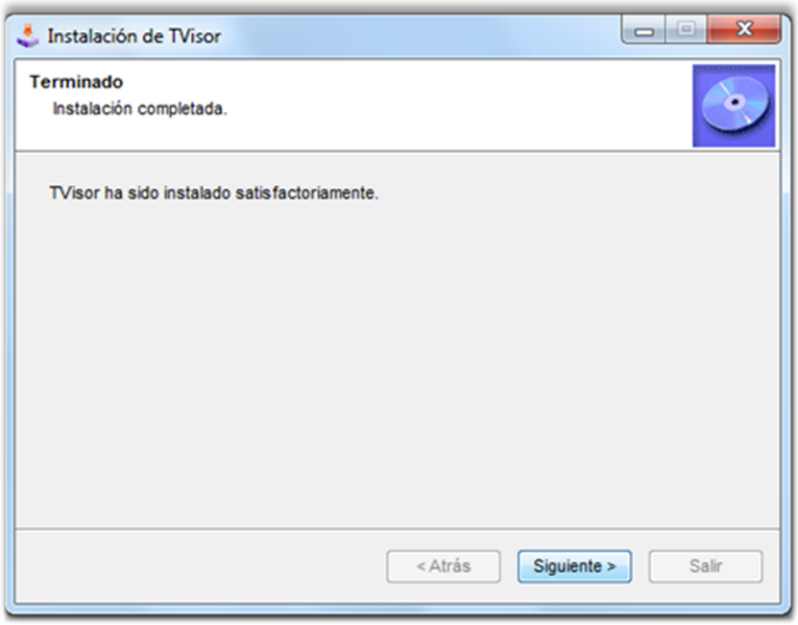
Paso 4: En esta ventana se puede seleccionar la locación de destino donde se desea instalar los archivos (se recomienda utilizar la ubicación que se muestra por defecto). Click “Siguiente” para continuar con la instalación.



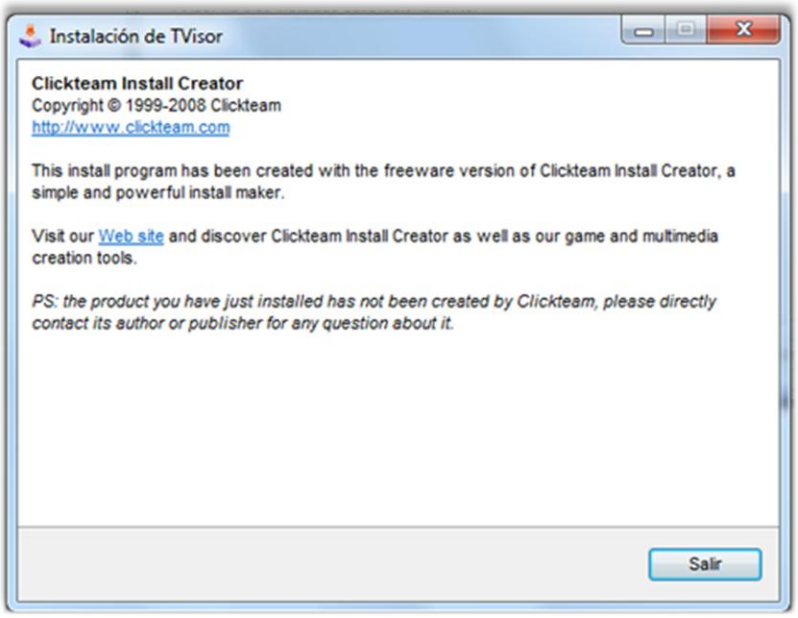
Paso 5: Click “Empezar” para instalar la aplicación.



Paso 6: Después que se haya terminado la instalación de TVisor en su PC, aparecerá un mensaje como el que se muestra a continuación. Click “Siguiente”.

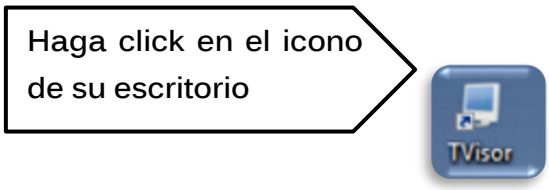


Paso 7: Click “Salir” para terminar la instalación.

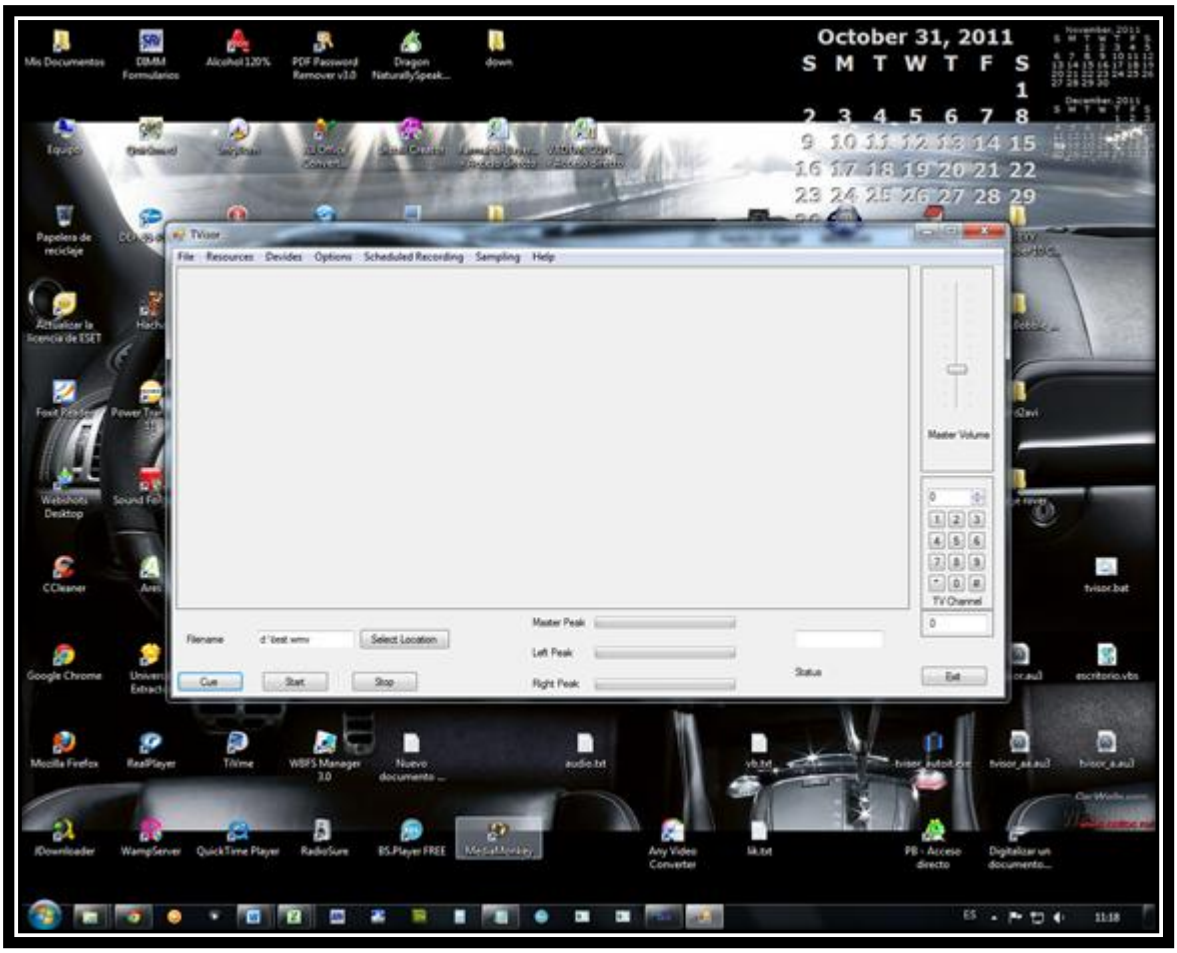


⇒ Uso de TVisor

→ Iniciando TVisor

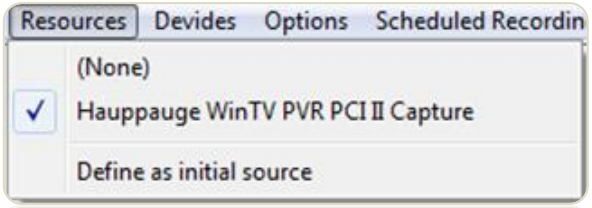


La Pantalla principal de la aplicación TVisor será mostrada.

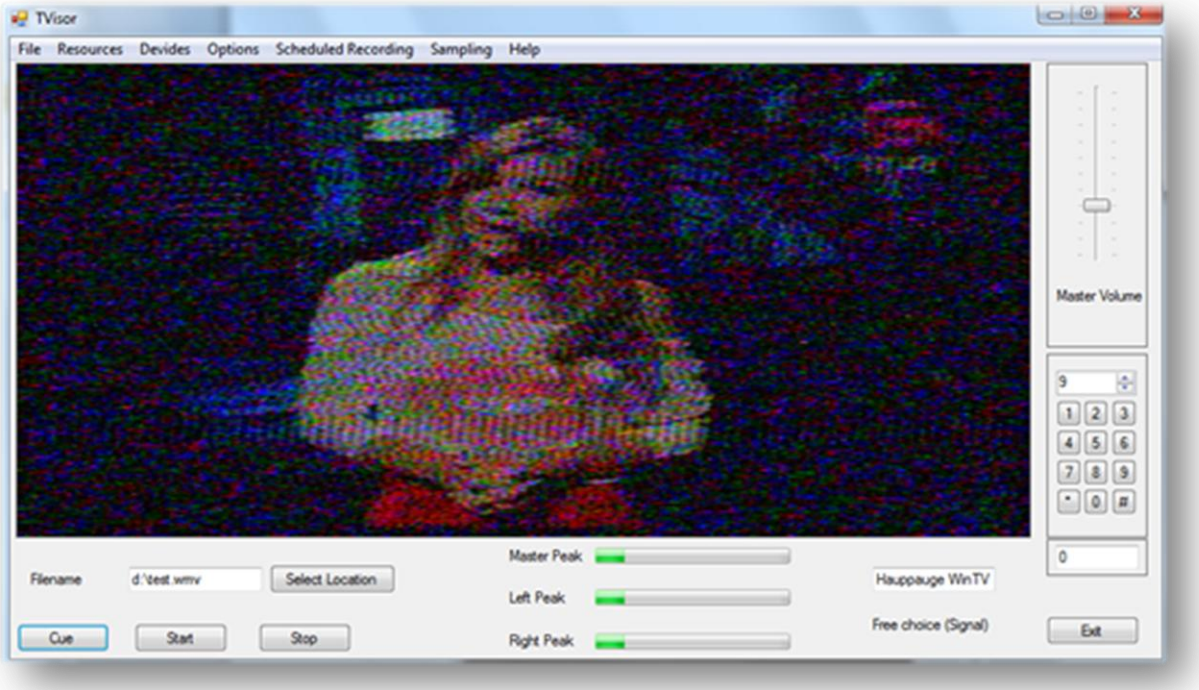


→ Selección del dispositivo de captura

En el menú “Resources” se puede seleccionar el dispositivo de entrada, para el caso la tarjeta sintonizadora de TV. La opción por defecto es None (ninguna). Como ejemplo se muestra la tarjeta Hauppauge WinTV PVR.



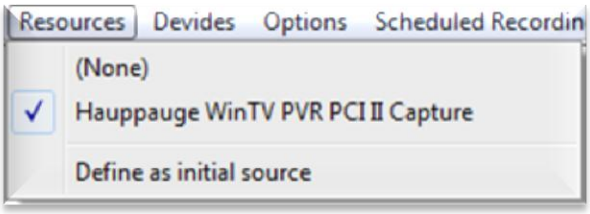
Una vez seleccionado el dispositivo de entrada aparece la pantalla principal de la aplicación.



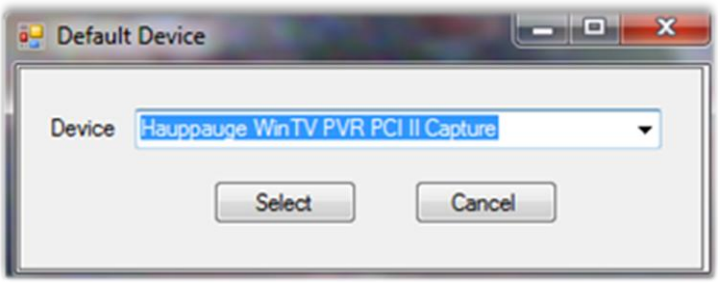
⇒ Funciones de la aplicación TVisor

⊕ Selección automática del dispositivo de captura

Para evitar la selección del dispositivo de captura al inicio de la aplicación. Es posible definirlo por defecto, a través del menú “Resources” submenú “Define as initial source”



Se selecciona el dispositivo de entrada por defecto.



⊕ Selección de canales

Permite sintonizar el canal deseado a través del uso de los botones o escribiendo el número en el cuadro superior.

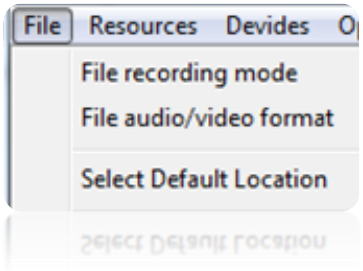
⊕ Control de volumen

Permite controlar el volumen de la aplicación.

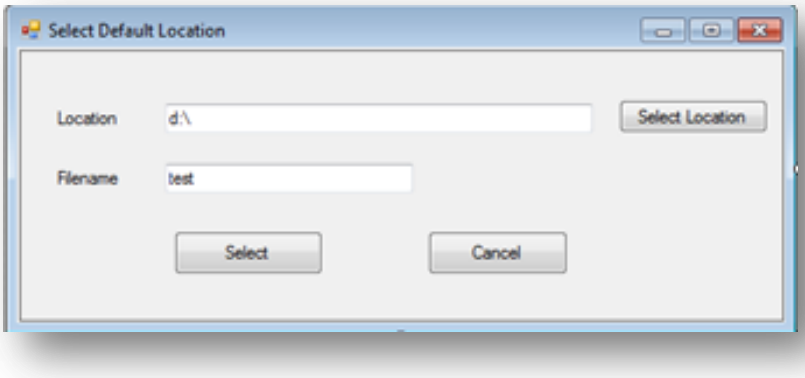
⊕ Grabación

Previamente es necesario seleccionar la ruta del archivo donde se va a grabar (por defecto será en el disco *"d:**"* bajo el nombre *"test.wmv"*).

Se puede modificar la ubicación y el nombre del archivo por defecto a través del menú "File" submenú "Select Default Location".



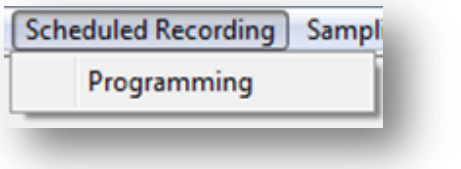
Se selecciona la ubicación en el disco duro y el nombre del archivo que se desee.



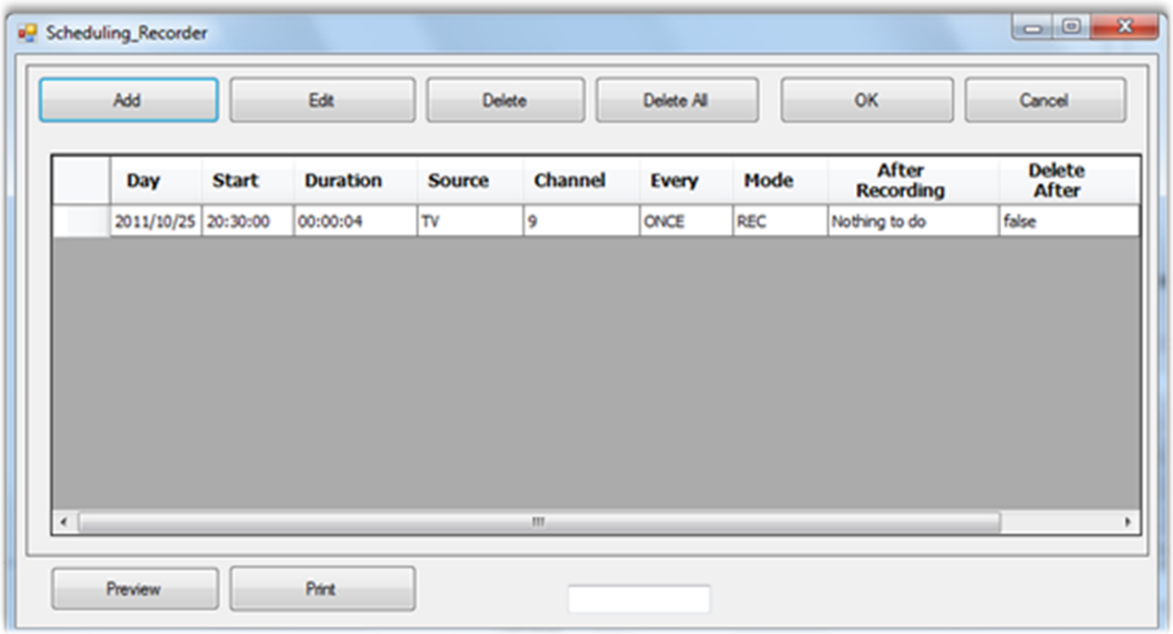
Haciendo click en el botón "Start" se inicia la captura del vídeo, guardando el archivo en la ubicación señalada. La captura termina al hacer click en el botón Stop.

⊕ Grabación programada

Se puede programar la grabación a través del menú “Scheduled Recording” submenú “Programming”.

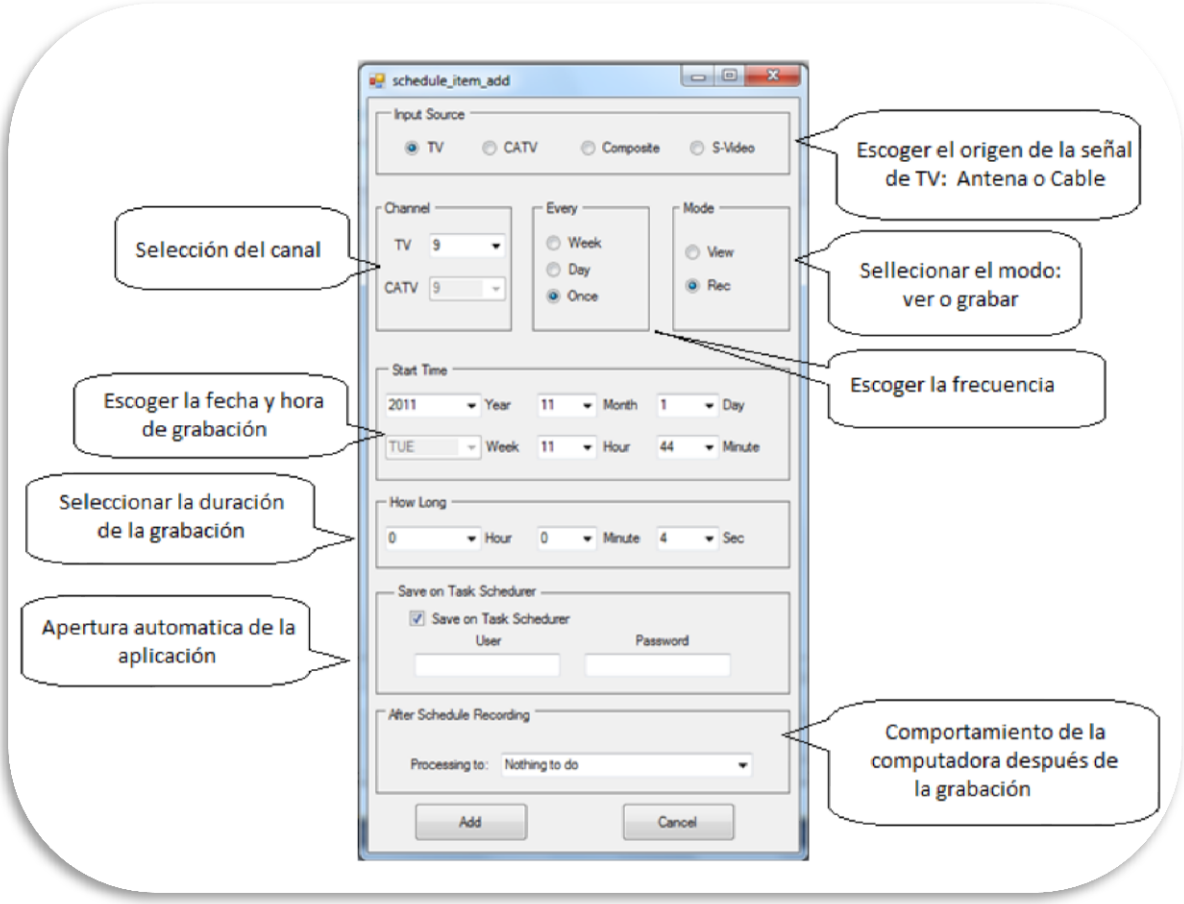


La pantalla muestra un resumen de la programación de la grabación.



Las opciones disponibles para la programación de la grabación son:

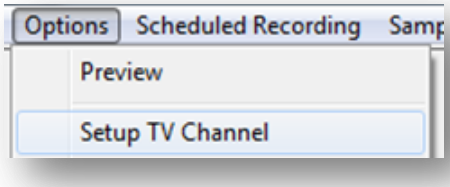
- **Add:** Permite añadir o crear un nuevo registro de grabación. Configurando las opciones de la siguiente pantalla:



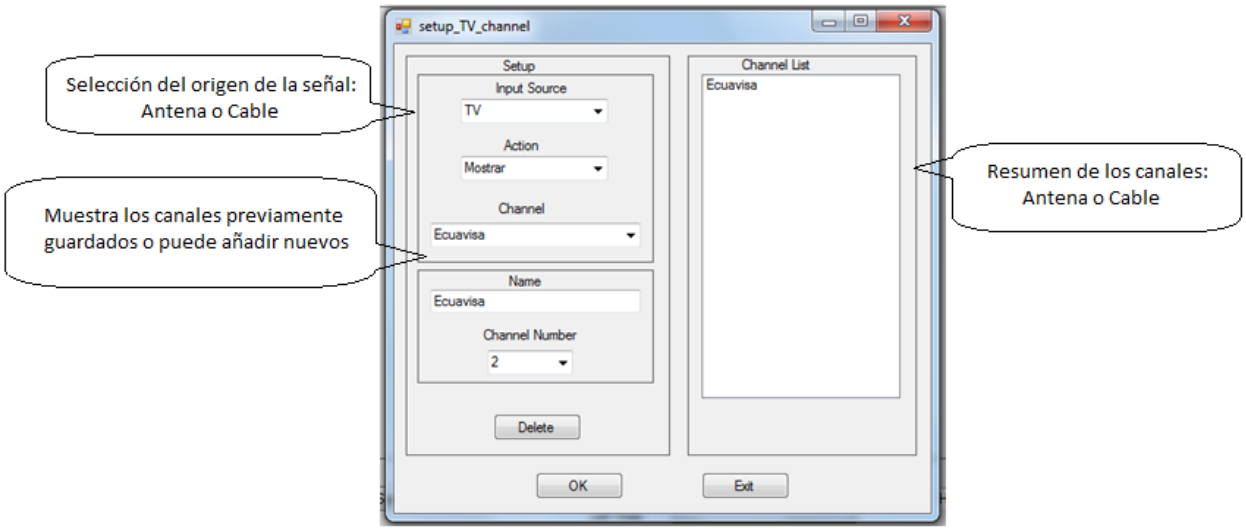
- **Edit:** Permite editar un registro de grabación.
- **Delete:** Permite eliminar un registro de grabación.
- **Delete All:** Permite eliminar todos los registros de grabación.
- **Print:** Permite imprimir los registros almacenados de grabación
- **Preview:** Permite visualizar como se imprimirán los registros almacenados de grabación

⊕ Configuración de los canales de TV

Se puede configurar los canales de TV a través del menú “Options” submenú “Setup TV Channel”.

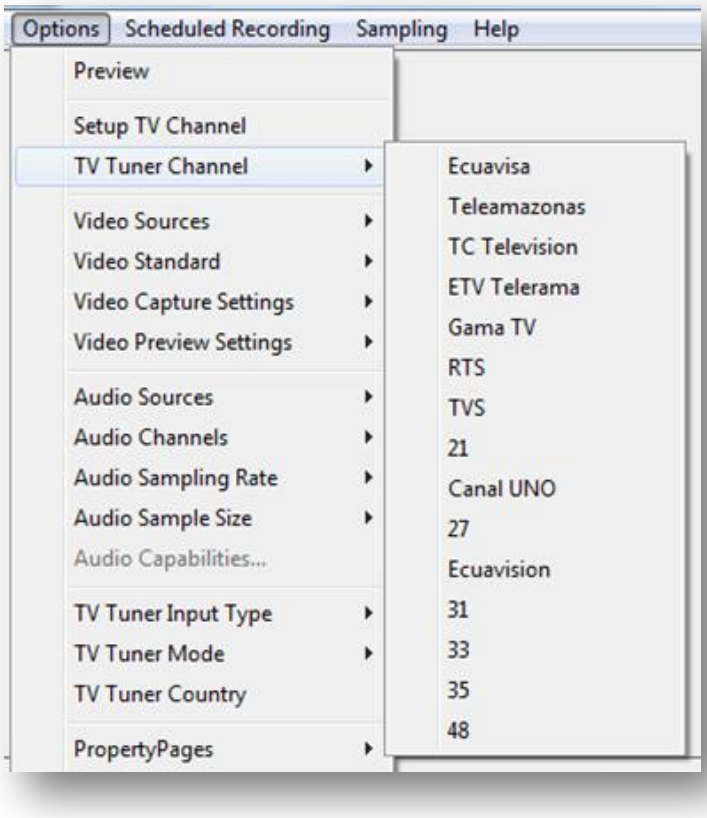


La pantalla muestra las opciones para la configuración de los canales.



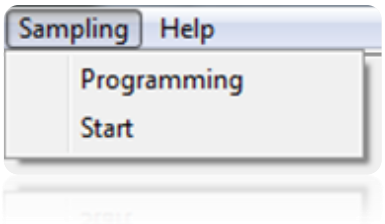
⊕ Lista de canales de señal abierta, almacenados en la base de datos

Se puede acceder a esta opción a través del menú “Options” submenú “TV Tuner Channel”.

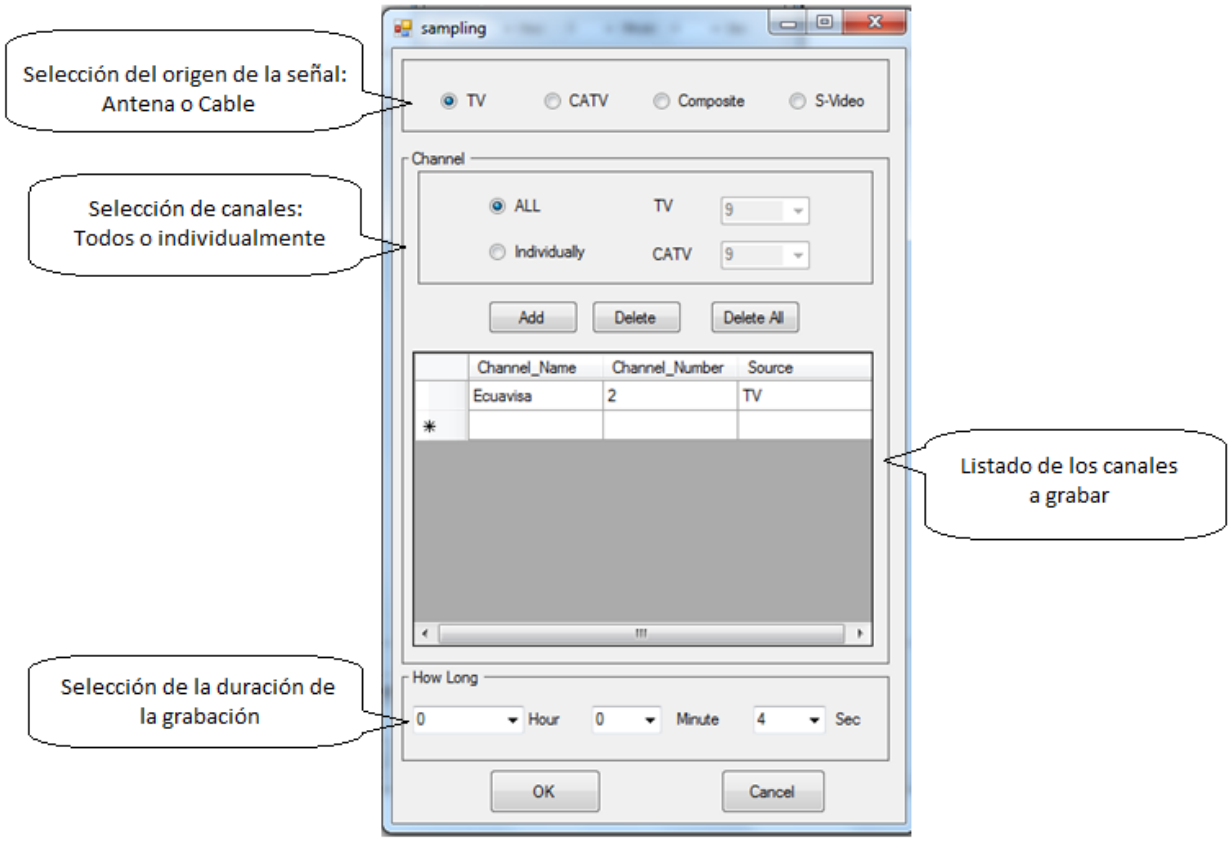


⊕ Muestreo de canales (Grabación)

Se puede grabar un muestreo de canales previamente almacenados en la base de datos o determinados canales con una duración igual para todos. La configuración de esta opción se accede a través del menú “Sampling” submenú “Programming”.



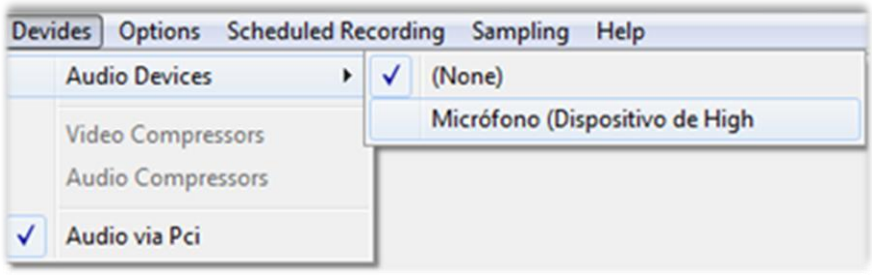
La siguiente pantalla muestra las opciones para la configuración de la grabación del muestreo de canales.



⊕ Configuración de los parámetros de audio y video

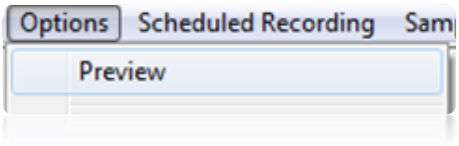
✦ Selección del dispositivo de ingreso de audio para la captura.

La opción por defecto es Audio via Pci, que es el audio proveniente de la tarjeta sintonizadora de TV.



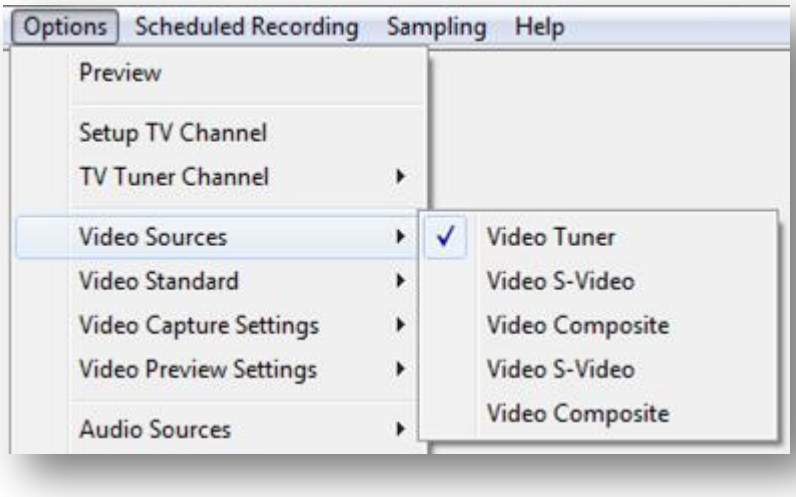
✦ Visualización en pantalla de la captura de vídeo.

Al seleccionar el dispositivo de captura, esta opción por defecto permite visualizar en pantalla la captura de vídeo.



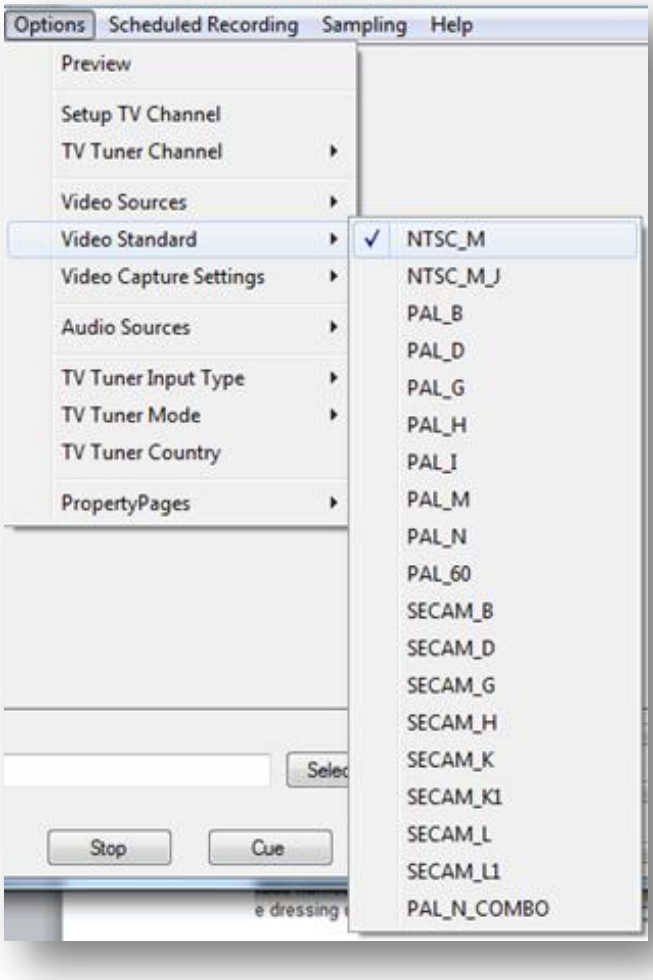
✦ Fuente de la señal del dispositivo de captura

Selecciona las entradas del dispositivo de captura. Por defecto es "Video Tuner" para tarjetas sintonizadoras de TV Pci.



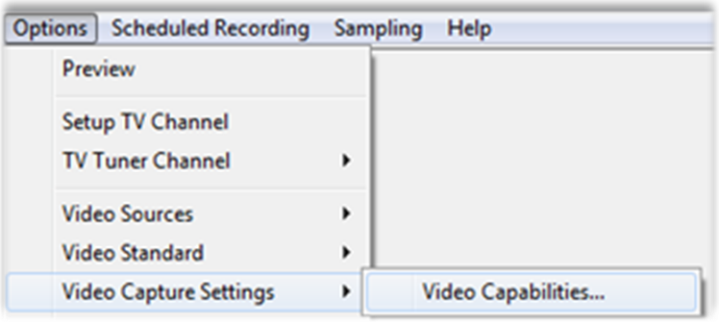
✦ Lista de estándares de televisión analógica

Muestra los estándares de televisión analógica soportados por la aplicación. Por defecto es NTSC_M.

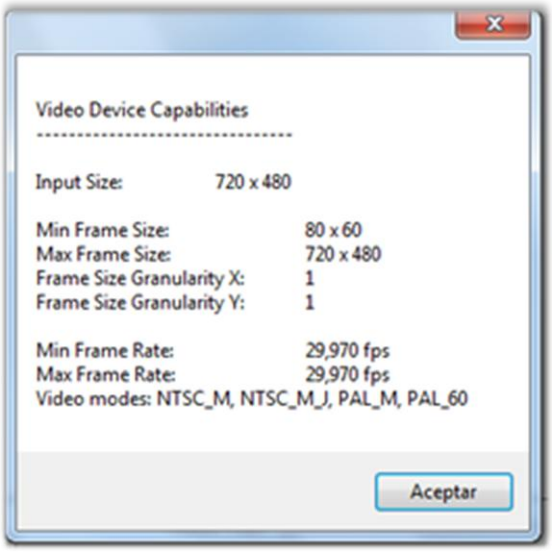


✦ Resumen de parámetros del vídeo capturado

Se puede acceder a través del menú “Options” submenú “Video Capture Settings”.

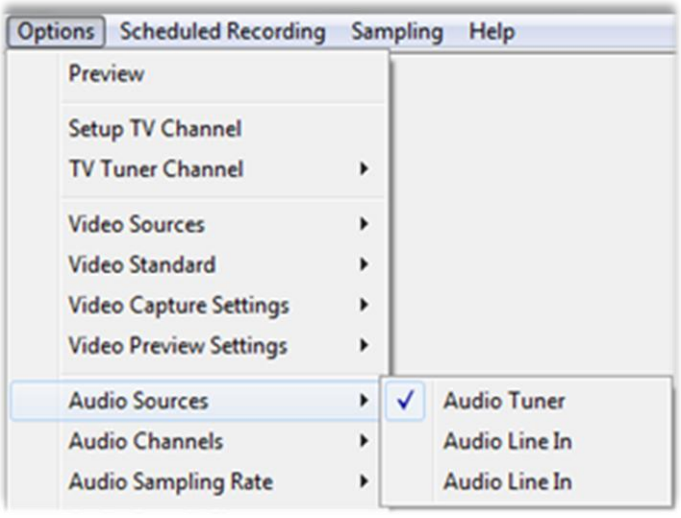


Muestra el resumen de los parámetros del vídeo como: tamaño de marco máximo / mínimo y la velocidad de fotogramas.



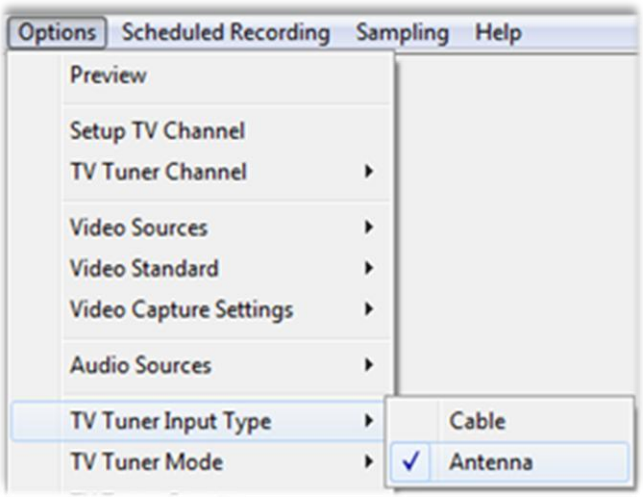
- **Fuente de la señal de audio del dispositivo de captura**

Selecciona las entradas de audio del dispositivo de captura. Por defecto es “Audio Tuner” para tarjetas sintonizadoras de TV Pci.



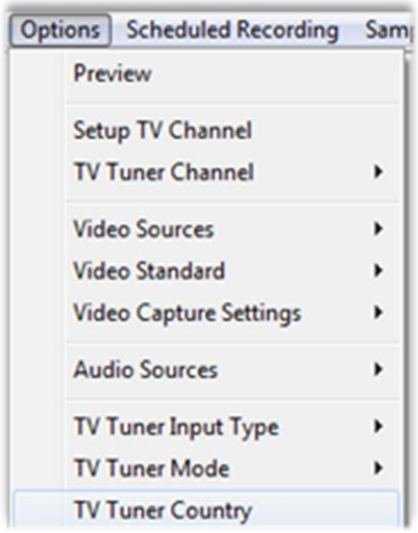
✦ Modos de recepción de la señal analógica

Por defecto es Antena.



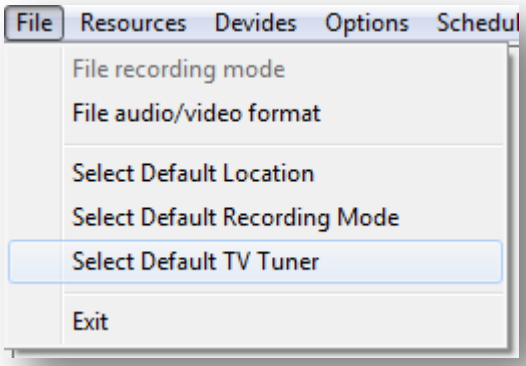
✦ Configuración de los parámetros de captura de vídeo

Se accede a esta opción a través del menú “Options” submenú “TV Tuner Country”.

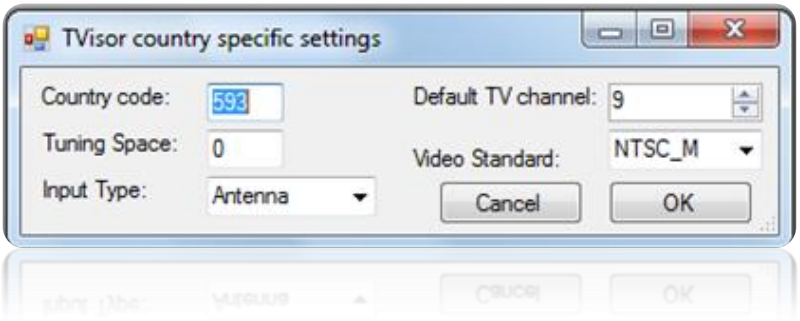


✦ Selección de las opciones de sintonización por defecto

Se accede a esta opción a través del menú “File” submenú “Select Default TV Tuner”.



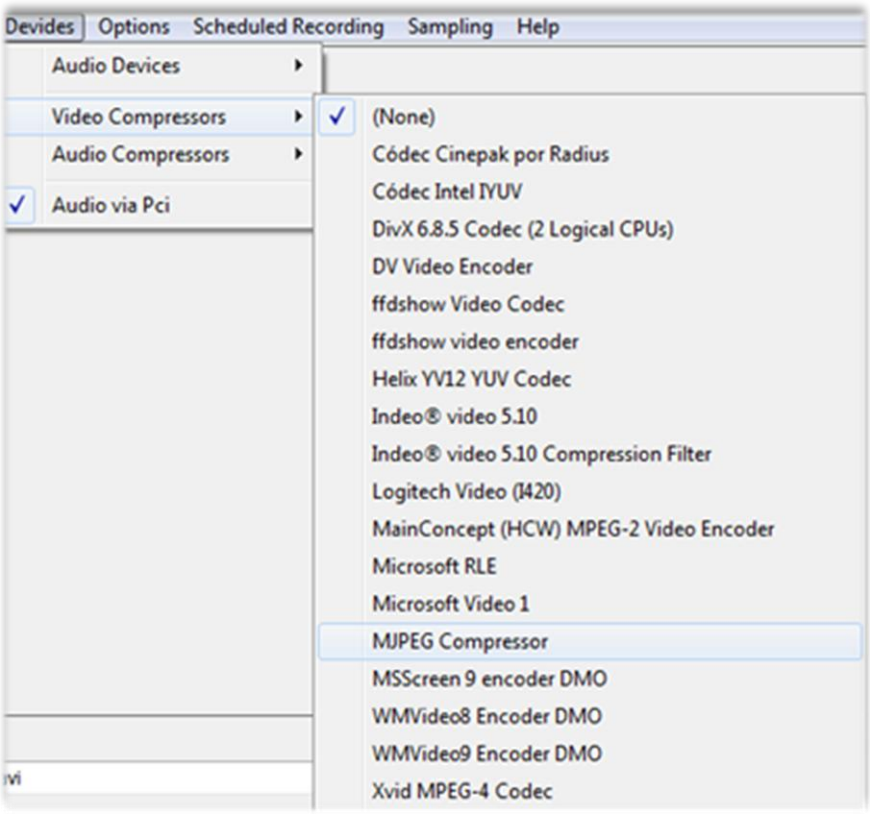
Una vez seleccionada esta opción permite seleccionar los siguientes parámetros por defecto:

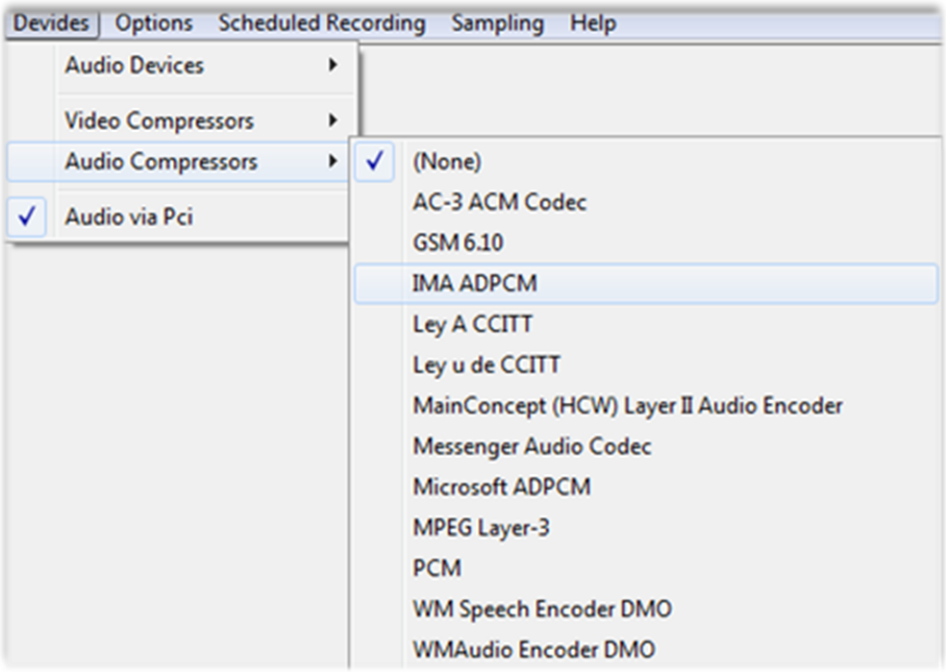


- **Country Code:** Dependerá del país donde se utilice. Para Ecuador está establecido el número 593.
- **Tunning Space:** Por defecto es 0.
- **Input Type:** Selección del tipo de entrada de la señal. Antena o cable.
- **Default TV cannel:** Es el canal por defecto que se sintonizará.
- **Video Standard:** Estándar de TV analógica que se puede utilizar. En el caso para Ecuador es NTSC_M.

✦ Modificación de los compresores de audio y vídeo

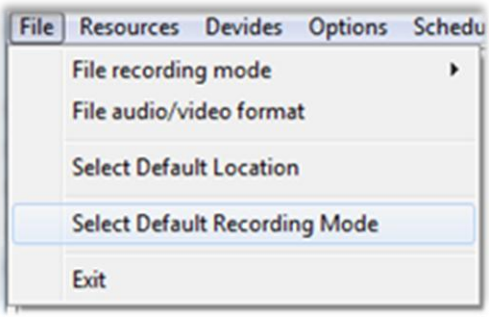
Permite escoger el formato de compresión de audio y vídeo.



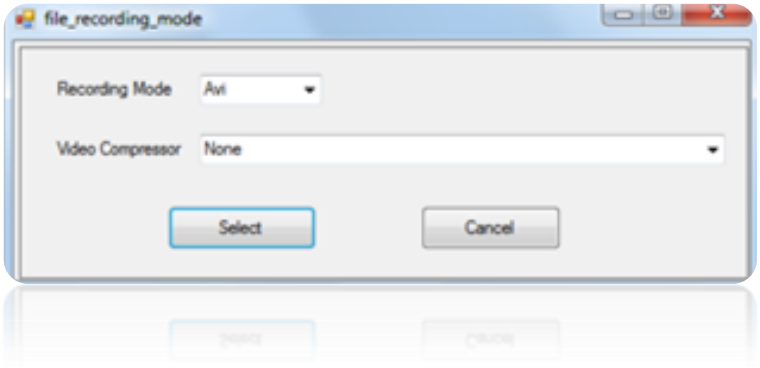


✦ Formato y Códec por defecto

Se accede a esta opción a través del menú “File” submenú “Select Default Recording Mode”.



Permite escoger el formato y el códec preferido al inicializar la aplicación, para utilizarlo por defecto.


file_recording_mode
Recording Mode
Avi
Video Compressor
None
Select
Cancel