UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sonido e Imagen)

"Estudio de la viabilidad sobre la integración del sistema de cámaras Robotizadas de la UPV-TV con el software Video Toaster "

TRABAJO FINAL DE CARRERA

Autor/es:
Javier Climent Sánchez

Director/es:
Antoni Josep Canós Martín
Ignacio Despujol Zabala

GANDIA, 2011

UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sonido e Imagen)

"Estudio de la viabilidad sobre la integración del sistema de cámaras robotizadas de la UPV-TV con el software Video Toaster "

TRABAJO FINAL DE CARRERA

Autor:

Javier Climent Sánchez

Directores:

Antoni Josep Canós Marín

Ignacio Despujol Zabala

UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

GANDIA. 2011

Índice

1. Introducción:

El presente proyecto principalmente consiste en realizar un estudio de viabilidad sobre la integración de un Pc con un sistema de cámaras robóticas Panasonic. Para ello, diseñaremos un software y un hardware que controlen los principales controles de operación y las funciones de las cámaras.

Como segundo objetivo sería integrar el sistema de cámaras con el software (Video Toaster) para poder automatizar algunas funciones de operación de las cámaras con los escenarios virtuales de la herramienta LiveSet. Esto reduciría las acciones de operación que debe realizar el técnico del estudio a la hora de realizar un programa de televisión.

El hardware nos permitirá conectarnos al sistema de cámaras, para comunicarnos de forma remota a través de un Pc. Por lo tanto, tendremos que determinar cuáles el mejor diseño de conexión entre los componentes del sistema de cámaras y el Pc.

Hay dos posibles diseños del hardware de comunicación que explicaremos y desarrollaremos más adelante. Uno sería del dispositivo "Remote Control Panel" (AW-RP505) a un "HUB" (AW-PR505, que es un panel de control híbrido múltiple de cámaras). Y el otro diseño sería de la cabeza robótica de cámara (AW-PH300A) directamente al Pc.

Para cada Hardware de conexión deberemos diseñar un software diferente, ya que cada uno se rige por un protocolo de comunicación distinto. De los cuales seleccionaremos el más adecuado, analizando sus limitaciones tanto de operatividad como conexionado.

Para ello, realizaremos las mediciones pertinentes en cada diseño de conexión, basadas en la captura de instrucciones sincronizadas entre los distintos dispositivos. De esta forma estudiaremos el protocolo de comunicación entre los mismos, consiguiendo programar métodos y funciones para el control de las cámaras desde el Pc.

Estas funciones y métodos permitirán el desarrollo de futuras aplicaciones informáticas y la integración de dispositivos externos al sistema de cámaras, por ejemplo:

- Controlar las cámaras por medio de un joystick, un controlador Midi, un dispositivo de control personalizado, etc.…

- Diseñar un software personalizado de escenarios virtuales 3D que interactué con los movimientos de posición, zoom y enfoque de cámara, es decir, por detección de movimiento por ejes de cámara.

- Integrar el sistema de cámaras a un programa de edición lineal como: Video Toaster. Este último ejemplo de aplicación con la Video Toaster, lo desarrollaremos más adelante en el estudio y analizaremos: su SDK, sus limitaciones y sus posibles aplicaciones al diseño de software.

En el aparatado: "2. Fundamentos del Sistema Pan/Tilt de cámaras robóticas Panasonic" hablaremos del funcionamiento, los dispositivos, los interfaces de conexión y las señales que conforman el sistema de cámaras, haciendo especial inca pié en la parte del sistema que controla las funciones de movimiento de las cámaras y despreciando la parte de las señales de video que se transmiten por medio de sistema.

En el apartado: "3. Análisis de la comunicación entre el panel de control y el Hub" explicaremos como realizaremos las medidas oportunas para analizar el protocolo de comunicación RS-422 que se encarga de la comunicación entre el panel de control de cámaras y el Hub Consola central de puertos múltiples: AW-HB505).

En el apartado: "4. Protocolos de Comunicación de la cabeza Pan/Tilt (AW-PH300A/500/600)" explicaremos detalladamente las características de la comunicación, los comandos que lo forman y su sintaxis. Y finalmente realizaremos una comparación de sus principales diferencias.

En el apartado: "5. Interface" diseñaremos un interface diferente para cada protocolo que nos permitirá conectarnos a las cámaras a través del PC, explicaremos sus principales características, los dispositivos que lo forman, mostraremos sus diferentes diagramas de conversión y compararemos sus principales características.

En el apartado: "6. Software" desarrollaremos un software para cada protocolo de comunicación y explicaremos el código detalladamente para cada versión de software, además de explicar sus principales características y su funcionamiento práctico. Uno de estos programas será el que elegiremos para conectarnos a las cámaras a través del PC.

En el apartado: "7. Futuras Aplicaciones" comentaremos posibles aplicaciones que se pueden desarrollar a partir de los métodos que hemos diseñado en el software.

En el apartado: "8. Integración del Sistema de cámaras con el software Video Toaster" explicaremos si podemos la idea integrar el control de las cámaras con el programa de edición lineal 'Video Toaster' y de no ser así explicaremos las limitaciones que hay que evitan el desarrollo de un software que integre el control de las cámaras con los escenarios virtuales de la video Toaster.

En el apartado: "9. Conclusiones" explicaremos el porqué la necesidad de realizar este estudio, los problemas que me he encontrado y las soluciones que he realizado.

En el apartado: "10. Videos" introduciremos unos videos de ejemplo en el cual mostraremos el funcionamiento del software elegido para cada función de control de las cámaras.

2. Fundamentos del Sistema Pan/Tilt de cámaras robóticas Panasonic

En este apartado vamos a explicar el funcionamiento del sistema, sus dispositivos y sus principales señales de control. También explicaremos las diversas señales del sistema entre dispositivos, analizaremos únicamente las señales de control excluyendo las señales de video y de sincronismos.

El sistema está formado por distintos dispositivos, nosotros nos centraremos únicamente en los principales: la cabeza Pan/Tilt, el HUB (Consola central de puertos múltiples) y el panel de control multi-híbrido.

En la siguiente imagen se muestra la configuración del conexionado del sistema de cámaras, sus diferentes dispositivos y sus señales:

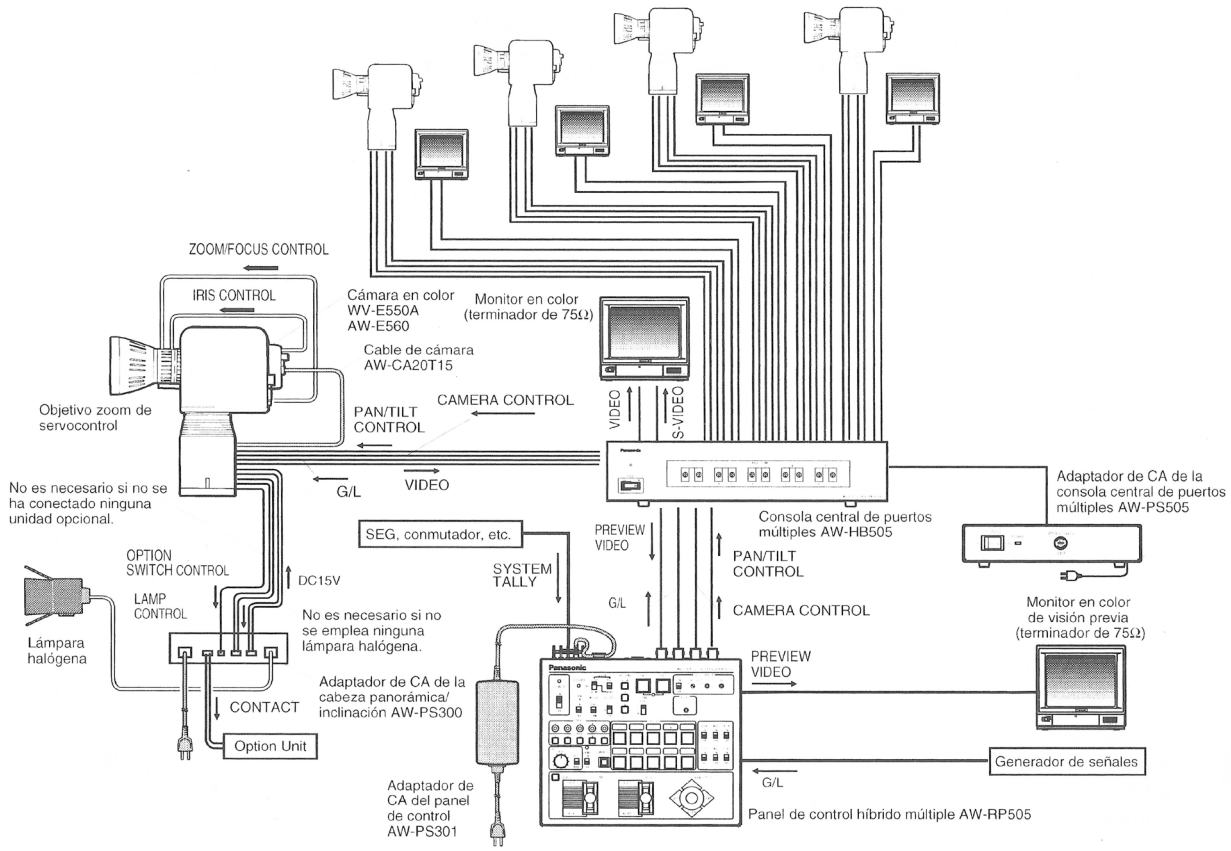
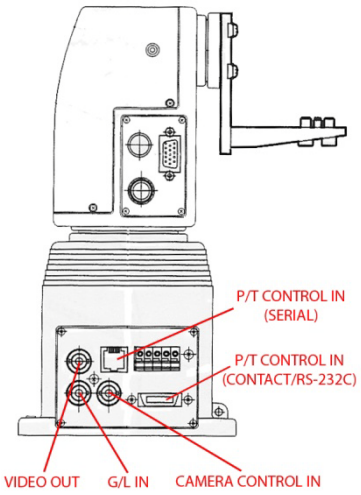


Fig. 1: "Diagrama de conexión del sistema de cámaras"

2.1. Dispositivos principales del sistema

2.1.1. Cabeza Pan/Tilt (AW-PH300A / AW-PH360L)

Es un posicionador de cámara panorámico y de inclinación que permite el control remoto de las cámaras y está provisto de un servo-control de Zoom, Foco e Iris. Además permite memorizar hasta un máximo 50 Pre-sets de ajustes de posición P/T y lentes por cabeza.



Su ángulo de inclinación es de 190º (±95º, arriba o abajo) y el ángulo de panoramización es de 300º (±150º, izquierda o derecha).

Tiene una velocidad máxima de panoramización de 25º/s y de inclinación de 20º/s.

La cabeza Pan/Tilt puede controlarse por medio del panel de control híbrido múltiple (AW-RP505) y a través de la consola central de puertos múltiple (AW-HB505), que podrá controlar hasta 5 cámaras y cabezas Pan/Tilt.

Las señales encargadas de controlar la cabeza de cámara desde la consola central de puertos múltiple son dos:

Señal de control de cámara: Es una señal que valida el cambio de cámara. Va por medio del conector "Camera Control Out" de la consola central de puertos múltiples a la cabeza Pan/Tilt "Camera Control IN". El cable es un coaxial BNC (5C-2V).

Señal de control de la cabeza Pan/Tilt: Es una señal que envía y recibe los comandos de control Pan/Tilt del panel por medio del "protocolo de comunicación RS422". Va conectado a la consola central de puertos múltiple "PAN/TILT Control Out (RJ-45)" hasta la cabeza PAN/TILT al conector "P/T Control In (RJ-45)". Por medio de un cable recto 10BASE-T (UTP categoría 5).

También es posible realizar el control simultáneo de cámaras y cabeza Pan/Tilt con un PC, a través del conector "P/T Control IN" (Rs-232C) de la cabeza de cámara. Este utiliza un protocolo distinto al del conector RJ-45 anterior, que es el "protocolo de comunicación RS232C". Sin embargo, la información del balance de blancos solo se puede establecer con el panel de control. Pero ganamos otros comandos que no existen en el protocolo RS422, como la obtención y establecimiento de la posición Pan/Tilt y del objetivo (Zoom, Focus e Iris) de la cámara, respecto a su posición inicial de cada uno.

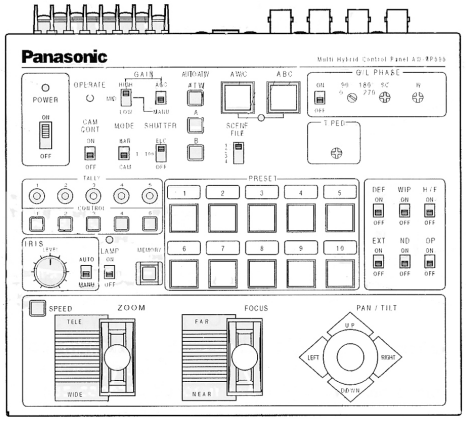
Cuando estén conectados el PC y el panel de control, y ambos envíen una operación en el mismo instante, tendrá prioridad la operación de control iniciada en último lugar del panel.

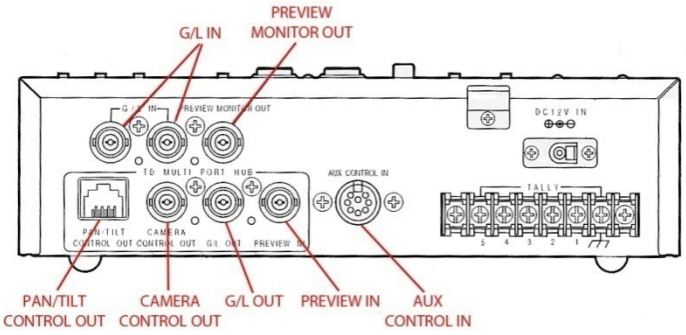
Este último punto lo trataremos más detenidamente cuando hablemos del Hardware de comunicación que diseñaremos.

2.1.2. Panel de control híbrido múltiple (AW-RP505)

Se combina con la consola central de puertos múltiples (AW-HB505) para controlar hasta 5 cabezas Pan/Tilt (AW-PH300).

El Panel de control puede controlar a través de las palancas de control: la inclinación y panoramización de las cabezas y los efectos de Zoom y enfoque. Y únicamente incorpora hasta 10 Pre-set de preajuste de posición P/T y Lentes por cabeza.





Este dispositivo efectúa diversas operaciones de control de las cámaras y las cabezas Pan/Tilt, emitiendo y recibiendo señales de video (PREVIEW IN), de control remoto (PAN/TILT CONTROL) y sincronización (G/L OUT).

El panel está conectado a la consola central por medio de 3 cables coaxiales (5C-2V): [PREVIEW IN] entrada de video previa, [G/L] salida de sincronización y [CAMERA CONTROL OUT] control de cámara. Y un cable recto de 10BASE-T (UTP categoría 5): [PAN/TILT CONTROL OUT], este último como hemos mencionado antes para el cabezal utiliza el mismo protocolo de comunicación RS422.

La longitud máxima de los cables desde este panel de control hasta la consola central es de 10 metros.

El panel de control posee un conector de entrada auxiliar para el control externo que únicamente incorpora las operaciones de panoramización, inclinación, enfoque, zoom e iris.

Funciones principales del Panel de Control (AW-RP505):

1. Selector de cabeza P/T de cámara.
2. Selector de la posición de pre-ajuste [PRE-SET] -> Solo hasta 10 Pre-Sets.
3. Palanca de Panoramización/inclinación, de Zoom y enfoque (FOCUS).
4. Control de Iris [IRIS LEVEL] y selector del modo de iris AUTOMATICO/MANUAL.
5. Interruptores de: Lámpara (LAMP), Deshelador (DEF), Limpiador (WIPE), Calefactor/Ventilador(H/F), Extensor del objetivo (EXT), Filtro ND (ND) y Opcional (OP)
6. Selectores de balance de blancos: ATW, Canal A y Canal B

2.1.3. Hub (Consola central de puertos múltiples: AW-HB505)

Cómo se puede observar en la imagen (**Fig. 1: "**Diagrama de conexión del sistema de cámaras"), la consola central se encarga de la comunicación entre el panel de control y las cabezas de cámaras. Además de gestionar y distribuir las señales de video, control Pan/Tilt y sincronismos. Es decir, actúa como un multiplexor y conmutador de señales del panel de control a los cabezales Pan/Tilt.

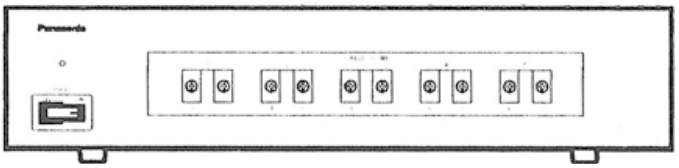


Fig. 2: "Hub (Consola central de puertos múltiples: AW-HB505)"

La consola central está conectada a cada cámara por medio de la cabeza Pan/Tilt, con 3 cable coaxiales (5C-2V): [PREVIEW IN] entrada de video previa, [G/L] salida de sincronización y [CAMERA CONTROL OUT] control de cámara. Y un cable recto de 10BASE-T (UTP categoría 5): [PAN/TILT CONTROL OUT]. Todas las conexiones se repiten para cada cámara.

La longitud máxima de los cables desde la consola central hasta la cabeza Pan/Tilt y las cámaras es de 500 metros.

2.2 Especificaciones del interface

2.2.1. Especificaciones del interface "PAN/TILT CONTROL RS-422":

Como hemos mencionado con anterioridad, la comunicación se realiza a través de un cable recto de 10BASE-T (UTP categoría 5).

El estándar utilizado para la comunicación entre dispositivos es la norma RS-422. Es similar a la norma RS-232, con la diferencia que el RS-422 usa una señalización balanceada o diferencial con Unidireccionalidad o no-reversible y puede soportar las topologías: Point-to-point, Multi-dropped.

Esta señalización ofrece la ventaja de mayor velocidad de transmisión de datos y una mayor distancia del medio físico (Cable). Para una velocidad de 100 Kbps la distancia del cable podrá llegar hasta los 1200 metros, en cambio conforme aumentamos la velocidad esta distancia disminuye, por ejemplo para la velocidad máxima que es de 10 Mbps podremos obtener una distancia máxima 12 metros. Esta característica hace que estas redes sean útiles en entornos industriales y aplicaciones similares.

En nuestro sistema de cámaras robóticas la velocidad de transmisión de datos será de 9600 bps, por lo tanto la distancia máxima será de 500m. Y la longitud de datos es de 7 bits con un bit de parada y paridad Par.

System	RS-422
Baud Rate	9600 bps
Bit Length	7 bits
Stop Bit	1 bit
Parity	Even
Flow Control	None

Este interface de control Pan/Tilt está formado por 3 buses, por los cuales se envían y se reciben los comandos de control Pan/Tilt utilizados por el "protocolo de comunicación Rs-422 Pan/Tilt".

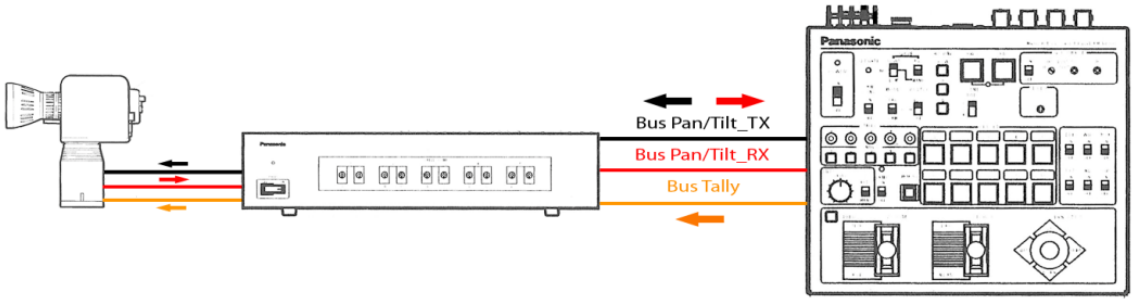


Fig. 3: "Diagrama de Buses del interface PAN/TILT CONTROL RS-422"

Bus de transmisión de comandos Pan/Tilt: Es el medio físico que transmite los comandos de operaciones enviadas por el Panel de control hibrido múltiple [AW-RP505] a la cabeza P/T.

Bus de Recepción de resultados Pan/Tilt: Es el encargado de enviar los resultados desde la cabeza de cámara Pan/Tilt al Panel de control múltiple hibrido a través del Hub [AW-HB505].

Bus de envío señal de Tally: A través de él se envía un comando desde el panel que indica a la Consola Central de puertos múltiples [AW-HB505], cuales la cámara seleccionada.

En la siguiente figura mostramos la configuración del conector "P/T CONTROL IN SERIAL":

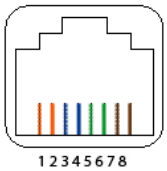


Fig. 4: "Conector P/T CONTROL IN SERIAL (RJ-45)"

Como se puede observar en la figura 4 es un conector RJ-45 y su configuración para cada pin es la siguiente:

Para el Panel de control híbrido múltiple:

PIN	SEÑAL DE CONTACTO	COLOR DEL CABLE	DESCRIPCION
1	No se usa	Naranja-Blanco	
2	No se usa	Naranja	
3	COLD 2	Azul-Blanco	Bus Pan/Tilt command -> Tx(-)
4	COLD 1	Azul	Bus Pan/Tilt command ->Rx(-)
5	HOT 1	Verde-Blanco	Bus Pan/Tilt command ->Rx(+)
6	HOT 2	Verde	Bus Pan/Tilt command ->Tx(+)
7	Tally input 1	Marrón-Blanco	Bus Tally-> Tx(-)
8	Tally input 2	Marrón	Bus Tally-> Tx(+)

Y para la Consola central de puertos múltiples, la dirección de la comunicación es inversa. Pero la configuración del cable con el conector es idéntica:

PIN	SEÑAL DE CONTACTO	COLOR DEL CABLE	DESCRIPCION
1	No se usa	Naranja-Blanco	
2	No se usa	Naranja	
3	COLD 2	Azul-Blanco	Bus Pan/Tilt command -> Rx(-)
4	COLD 1	Azul	Bus Pan/Tilt command ->Tx(-)
5	HOT 1	Verde-Blanco	Bus Pan/Tilt command ->Tx(+)
6	HOT 2	Verde	Bus Pan/Tilt command ->Rx(+)
7	Tally input 1	Marrón-Blanco	Bus Tally-> Rx(-)
8	Tally input 2	Marrón	Bus Tally-> Rx(+)

2.2.2. Especificaciones del interface “PAN/TILT CONTROL IN RS-232C”

El estándar utilizado para la comunicación entre la cabeza Pan/Tilt de cámaras y el PC es la norma RS-232C.

La interfaz RS-232 está diseñada para distancias cortas, hasta 15 metros según la norma, y para velocidades de comunicación bajas, de no más de 20 Kbits/segundo. A pesar de ello, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal Simplex, Half-Duplex o Full-Duplex.

En nuestro caso la velocidad de transmisión de datos será de 9600 bps, con una longitud de 8 bits y un bit de parada. La principal diferencia al interface anterior es que no posee paridad.

System	RS-232C
Baud Rate	9600 bps
Bit Length	8 bits
Stop Bit	1 bit
Parity	None
Flow Control	None

Este interface de control Pan/Tilt está formado por 2 buses, por los cuales se envían y se reciben los comandos de control Pan/Tilt utilizados por el “protocolo de comunicación Rs-232C Pan/Tilt”. Y un terminal estará conectado a tierra (GND).

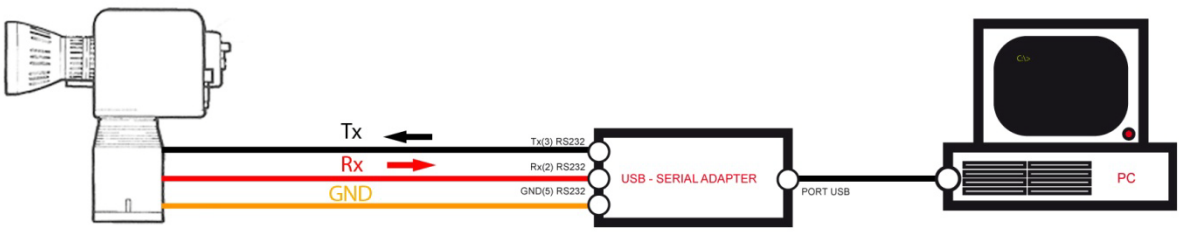


Fig. 5: “Diagrama de Buses del interface PAN/TILT CONTROL IN RS-232C”

En la siguiente figura mostramos la configuración del conector “PAN/TILT CONTROL IN RS-232C” de la cabeza P/T:

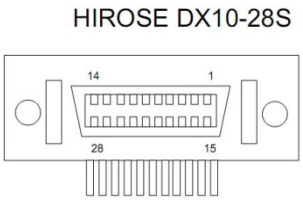


Fig. 6: “Conector PAN/TILT CONTROL IN RS-232C”

La comunicación de los dispositivos se realizará a través de un medio un cable que puede ser comprado (AW-CA28T9), pero es demasiado caro. También podemos hacerlo nosotros, que sale bastante más económico, veremos cómo hacerlo en el apartado “5.2. Interface RS232C: De la Cabeza P/T al PC”

En la siguiente tabla mostramos la configuración de los pines del conector “P/T Control IN (CONTACT/RS-232C)”:

Nº de Patilla	Señal de Contacto	Señal de RS-232C	Funciones
1-7	No se usa	No se usa	----------
8	No se usa	TXD	Datos de Tx de la cabeza Pan/Tilt.
9	No se usa	RXD	Datos de Rx de la cabeza Pan/Tilt.
10	GND	GND	----------
11	UP	No se usa	Entrada Control Tilt hacia arriba.
12	DOWN	No se usa	Entrada Control Tilt hacia abajo.
13	LEFT	No se usa	Entrada Control Pan hacia la izquierda.
14	RIGHT	No se usa	Entrada Control Pan hacia la derecha.
15	RUN-SING	No se usa	Señal de salida que indica el movimiento de la cabeza Pan/Tilt.
16-25	No se usa	No se usa	----------
26	DI OPT SEL	No se usa	Entrada de control de alimentación de la cabeza Pan/Tilt y de la cámara.
27	No se usa	No se usa	----------
28	GND	GND	----------

3. Análisis de la comunicación entre el panel de control y el Hub:

Únicamente realizaremos medidas para el interface RS-422 entre el panel de control y el Hub. Porque como veremos más adelante la comunicación entre ambos es constante, y siempre está enviando y recibiendo una secuencia de comandos. Por ello debemos tomar y analizar medidas para descifrar las diferentes secuencias de comandos para cada una de sus funciones.

En cambio, para el interface RS-232 entre el PC y el cabezal P/T no necesitaremos realizar medidas porque su protocolo de comunicación RS-232C está basado en eventos, es decir no está constantemente enviando comandos, únicamente envía uno para cada función.

Para la realización del análisis de las señales de control Pan/Tilt utilizaremos un software de monitorización de puertos RS-232 llamado "COM Port Tool kit v3.9".

Este software nos permitirá captar los comandos de control Pan/Tilt para cada instante de tiempo. De esta forma, podremos analizar los distintos protocolos de comunicación entre dispositivos y comprobar el envío y recepción de datos entre el PC y el dispositivo que deseamos controlar.

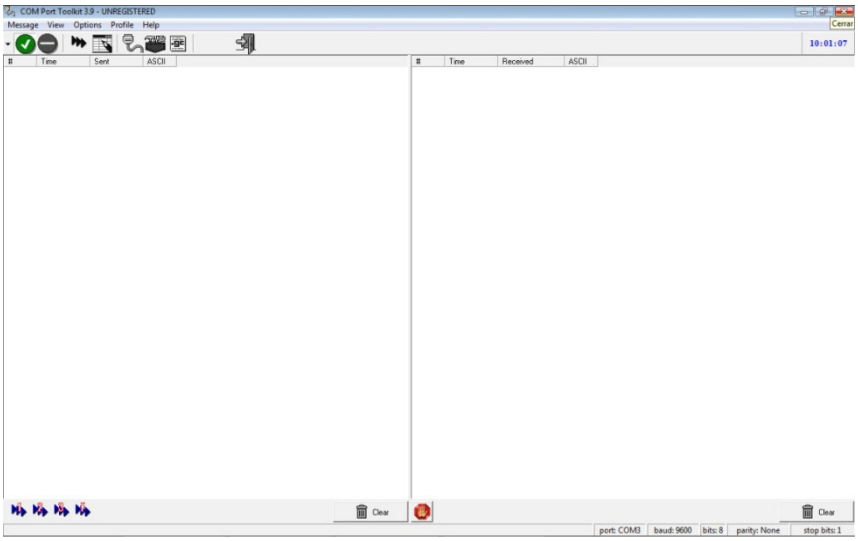


Fig. 1: "Aplicación COM Port ToolKit"

Directorio de la instalación 'COM PortToolKit': ...\Ficheros anexos\Software y cabeceras\Software para la monitorización y el envío de Puertos Serie (versión demo)

Para realizar el análisis de la comunicación entre "la consola central de puertos múltiples (AW-HB505)" y "el Panel de control híbrido múltiple (AW-RP505)", lo primero que debemos hacer para poder realizar las medidas es el diseño de un circuito que nos permita captar, con el PC, las señales de control entre estos dispositivos, las cuales se transmiten por medio del cable recto de 10BASE-T (UTP categoría 5). Este cable está conectado a estos dispositivos por el conector de entrada/salida "PAN/TILT CONTROL" de los mismos.

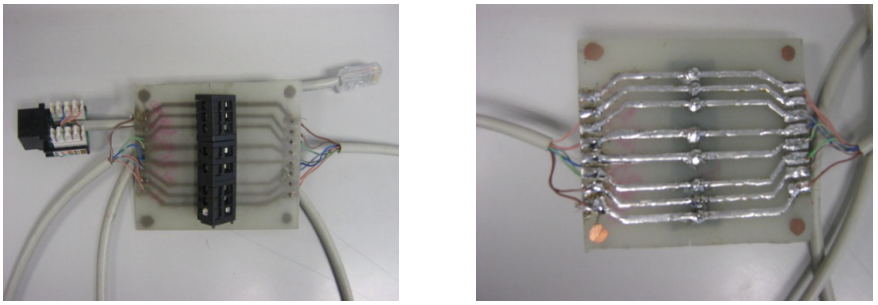


Fig. 2: "Imágenes del circuito impreso para la captación de medidas"

Esta interfaz utiliza "el protocolo de comunicación PAN/TILT Rs-422". Por lo tanto para monitorizar los tres BUSES de estos dispositivos a la vez, necesitaremos tres conversores RS-422 a RS-232 y tres adaptadores UBS a puerto serie RS232 conectados al PC para cada uno de BUSES.

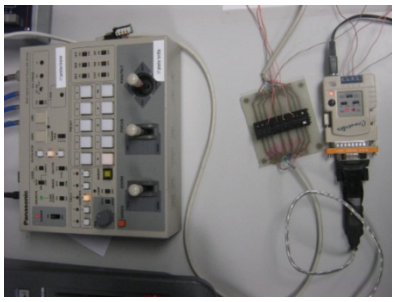


Fig. 3: "Imagen de conexión de la interfaz para la captación de medidas"

De esta forma, analizaremos el protocolo de comunicación entre **"**el Panel de control híbrido múltiple (AW-RP505)" y "la consola central de puertos múltiples (AW-HB505)", captando los comandos enviados entre los mismos para cada instante de tiempo y función del panel.

Cada uno de los conversores lo configuraremos en modo monitor 'MON' y en equipo terminal de datos 'DTE'. Así el conversor ajusta todos sus terminales i/o en modo recepción, es decir en modo escucha.

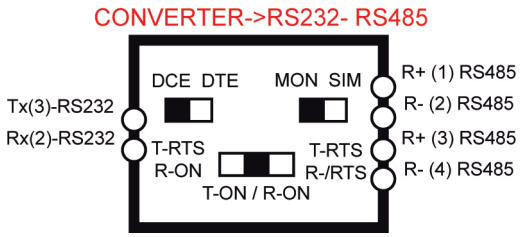


Fig. 4: "Configuración del conversor i485"

Conectaremos cada conversor a un bus, según se muestra en la siguiente tabla:

COLOR DEL CABLE	DESCRIPCION	CONVERSOR		BUS
Verde	Bus command Pan/Tilt->Tx(+)	Conversor 1	R1+(1)	Pan/Tilt command TX
Azul-Blanco	Bus command Pan/Tilt-> Tx(-)		R1-(2)	
Verde-Blanco	Bus command Pan/Tilt->Rx(+)	Conversor 2	R1+(1)	Pan/Tilt command RX
Azul	Bus command Pan/Tilt->Rx(-)		R1-(2)	
Marrón	Bus Tally-> Tx(+)	Conversor 3	R1+(1)	Tally
Marrón-Blanco	Bus Tally-> Tx(-)		R1-(2)	

En el siguiente diagrama se muestra cómo será el montaje completo del sistema de medición:

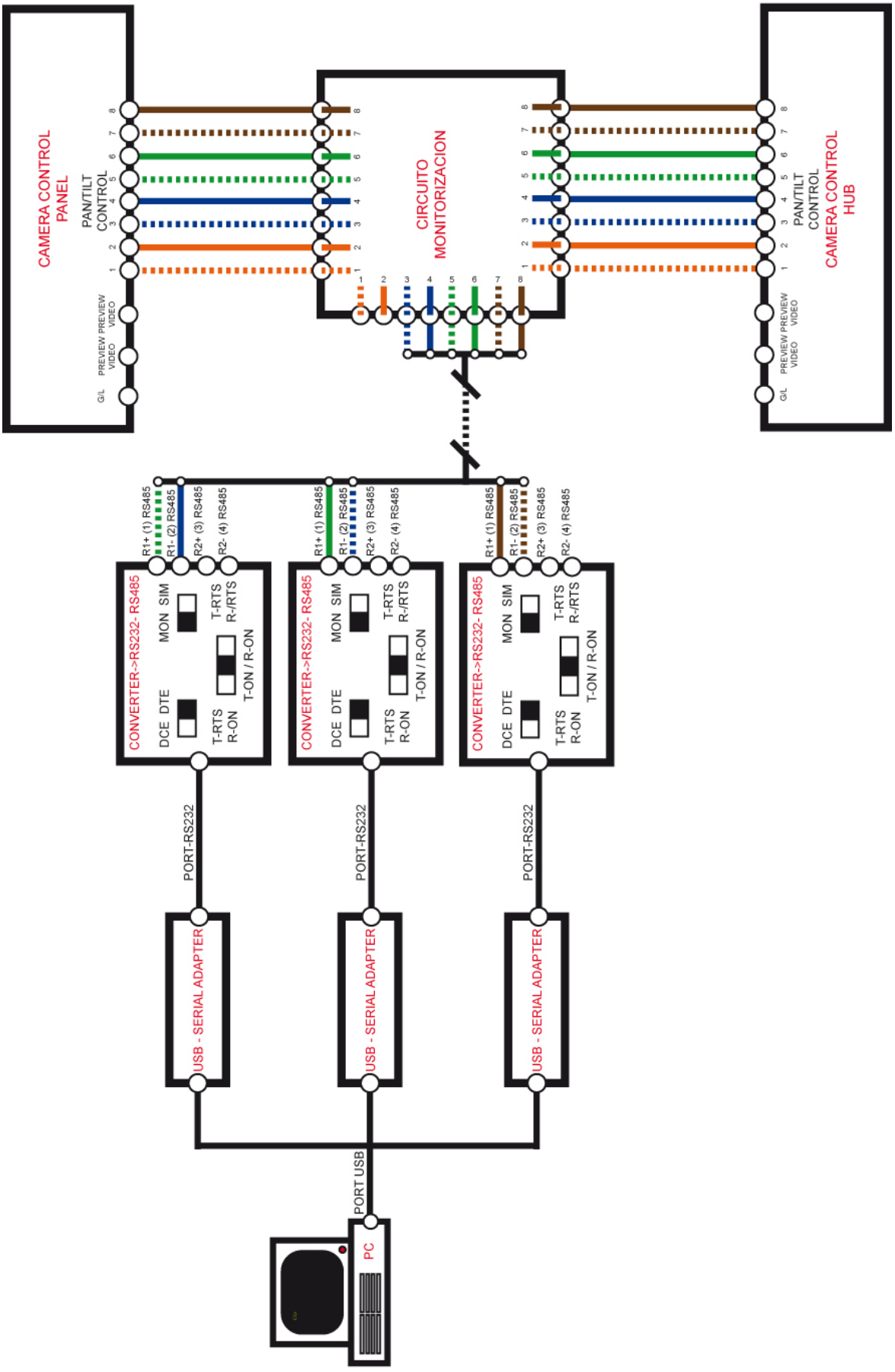


Fig. 5: "Diagrama de conexión de la interfaz de medida"

3.1. Procedimiento de medida:

Realizaremos las mediciones pertinentes para cada función del panel, basadas en la captura sincronizada de secuencias de comandos para los 3 buses: Bus de Tx P/T, Bus de Rx P/T y Bus de Tally.

Procedimiento:

1. Abriremos 3 veces el programa "COM Port Tool kit v3.9", para poder capturar los datos de los 3 Buses en el mismo margen de tiempo.

2. Seleccionaremos un puerto diferente para cada uno y estableceremos los siguientes ajustes de puerto ("Options → COM Port Configuration")  , para cada unos de los tres.

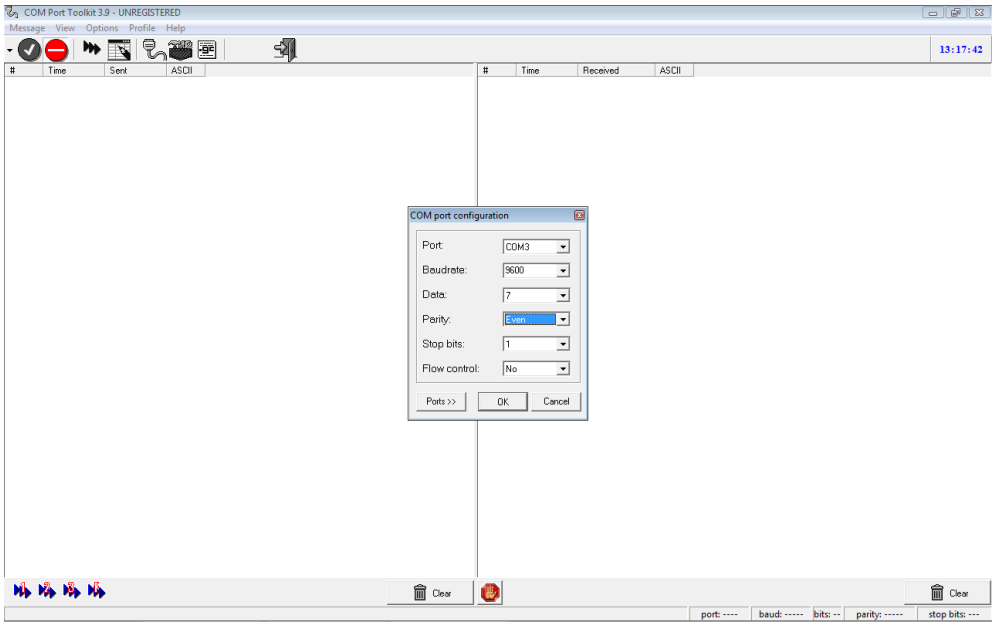


Fig. 6: "Ventana de configuración del Puerto serie"

System	RS-232
Baud Rate	9600 bps
Bit Length	7 bits
Stop Bit	1 bit
Parity	Even
Flow Control	None

Tabla de ajustes del puerto RS-232

3. Establecer las preferencias de captura de datos, es decir cómo se mostrará la longitud en bytes de los datos recibidos (Options→Preferences→Terminal/Listen). 

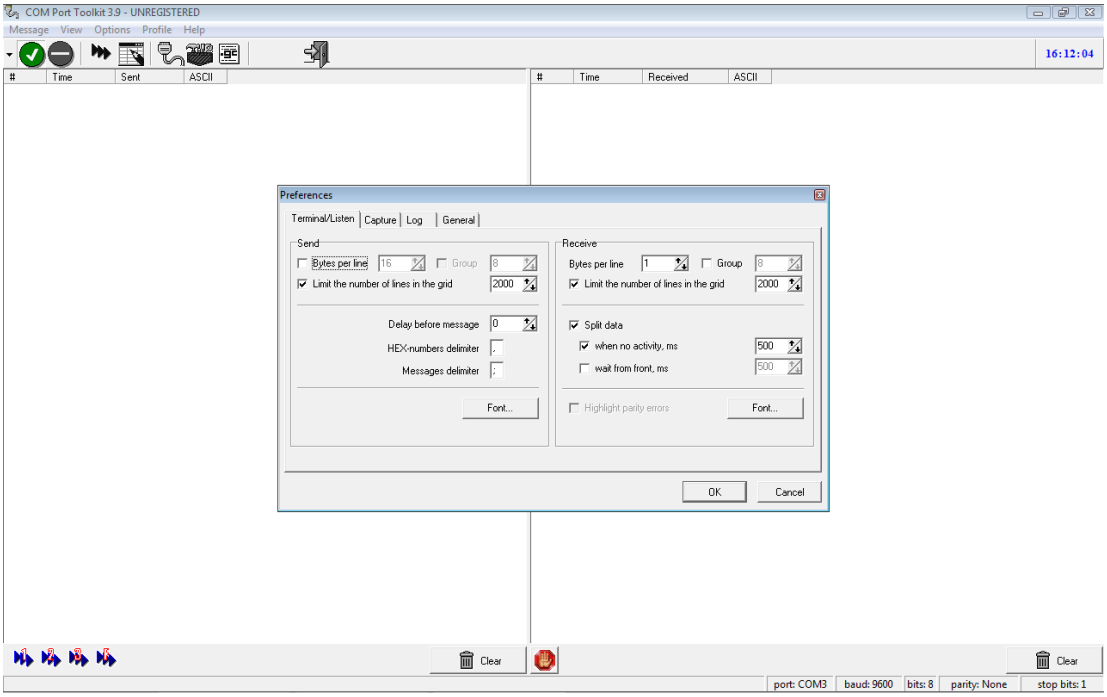


Fig.7: "Ventana de ajustes de Preferencias"

En el recuadro de 'Recive', introducimos en la variable 'Bytes per line'=1.

En la pestaña 'Capture', en el recuadro View en la misma variable introducimos el valor 'Bytes per line'=1 (Options→Preferences→Capture). Esto no servirá para visualizar los datos obtenidos byte a byte.

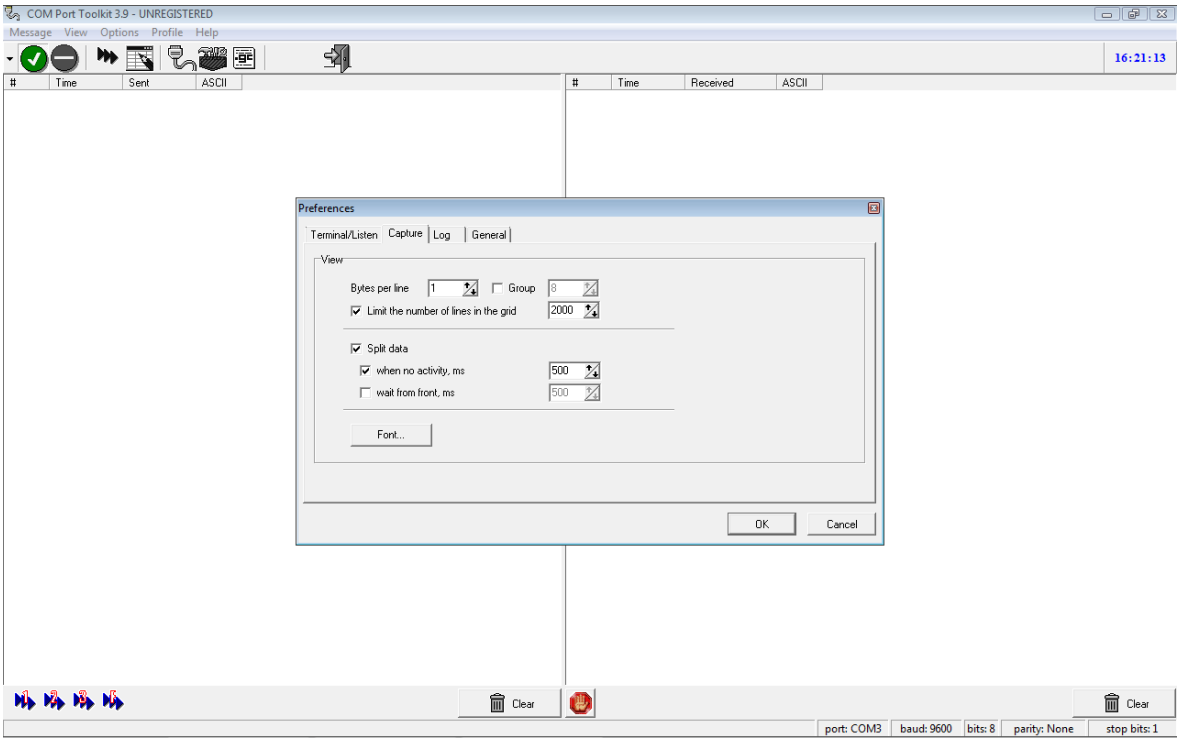


Fig.8: "Ventana de ajustes de Preferencias, pestaña Capture"

4. También introduciremos el directorio " ..\captura de Buses*.dat" para guardar los archivos con las medidas (Options→Preferences→Log).

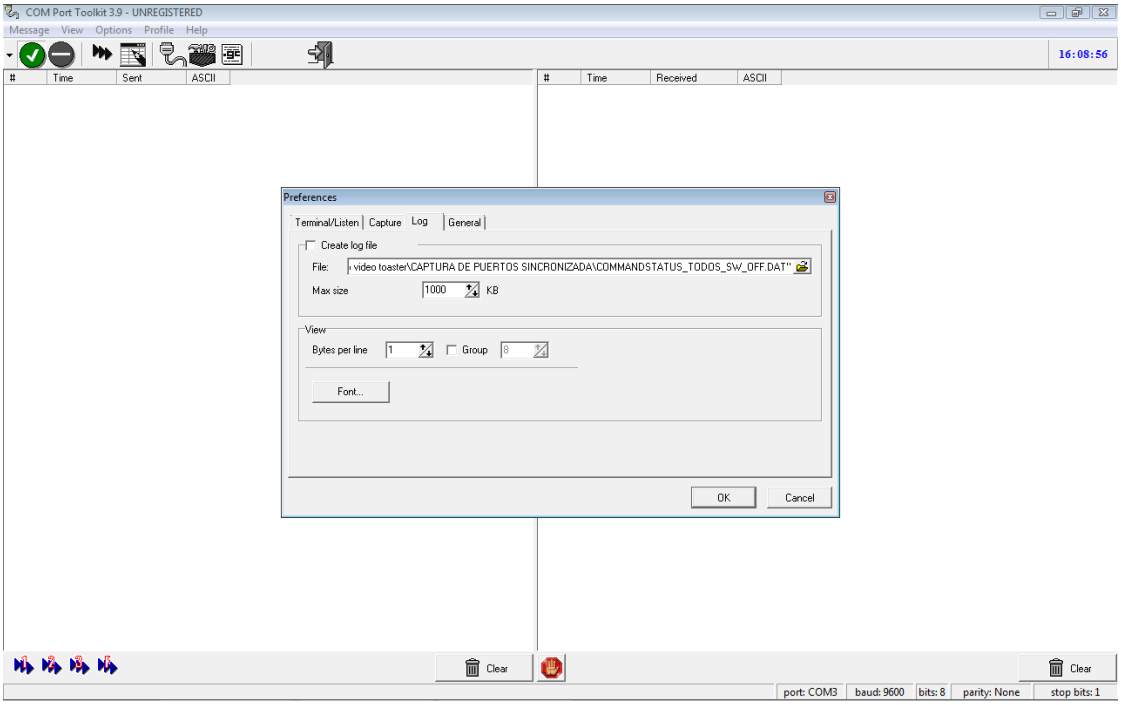


Fig.9: "Ventana de ajustes de Preferencias, pestaña Log"

5. Y Finalmente, ejecutamos para los 3 programas, la captura de mediadas presionado el botón 'Open Port'. Para pausar la monitorización utilizaremos el botón 'Recive On/Off'.

3.2. Tablas de Mediciones:

En este apartado hemos colocado las medidas más relevantes, si se desea ver el conjunto completo de medidas, mirar en la carpeta <..\Ficheros anexos\tablas de mediciones\> del DVD. En los archivos de Excel están las medidas "*.xlsx". También puedes pulsar en los hipervínculos que hay al final de cada apartado.

Comando de estado (Status Command):

CAM1				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	500	500		E0.H0.			45 30 0D 48 30 0D	
0	47	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
47	94	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
94	141	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
141	187	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
187	234	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
234	281	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
281	328	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
328	375	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
375	422	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
422	469	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
469	516	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
500	1000	500		E0.H0.			45 30 0D 48 30 0D	

CAM2				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	250	250		H0.E0.			48 30 0D 45 30 0D	
0	47	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
32	78	46			|			7C
47	94	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
78	141	63			|			7C
94	141	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
141	188	47	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
188	235	47	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
235	297	62	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
235	282	47			|			7C
250	500	250		H0.E0.			48 30 0D 45 30 0D	

CAM3				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	485	485		E0.H0.			45 30 0D 48 30 0D	
0	47	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
32	79	47			|			7C
47	94	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
79	125	46			|			7C
94	141	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
125	172	47			|			7C
141	188	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
172	219	47			|			7C
188	235	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
219	282	63			|			7C
235	282	47	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
282	344	62	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
344	391	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
375	422	47			|			7C
391	438	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
422	469	47			|			7C
438	485	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
469	516	47			|			7C
485	547	62	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
485	985	500		E0.H0.			45 30 0D 48 30 0D	

CAM4				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	500	500		E0.H0.			45 30 0D 48 30 0D	
15	62	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
15	62	47			`			60
62	128	66	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
62	116	54			`			60
116	163	47			`			60
128	175	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
163	210	47			`			60
175	222	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
210	274	64			`			60
222	268	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
268	315	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
274	319	45			`			60
315	362	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
319	366	47			`			60
362	425	63	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
366	413	47			`			60
413	460	47			`			60
425	500	75	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
460	507	47			`			60
500	546	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
500	1000	500		E0.H0.			45 30 0D 48 30 0D	

Archivo Excel: ..\Ficheros anexos\tablas de mediciones\Tabla de medidas StatusCommand_panel a Hub.xlsx

Cambio de Cámara:

CAM2-CAM1				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	188	188		E0.H0.			45 30 0D 48 30 0D	
16	78	62			|			7C
31	78	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
78	125	47			|			7C
78	125	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
125	172	47			|			7C
125	172	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
172	172	0			|			7C
172	188	16	Z****.			5A 2A 2A 2A 2A 0D		
188	250	62		z.			7A 0D	
188	391	203	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
250	484	234		E0.H0.			45 30 0D 48 30 0D	
391	438	47	X0****. X0****.			58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		
438	484	46	X0****. X0****.			58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		
484	547	63		s5105.			73 35 31 30 35 0D	
484	500	16	S****.			53 2A 2A 2A 2A 0D		
500	547	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
547	1047	500		E0.H0.			45 30 0D 48 30 0D	
547	594	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		

CAM1-CAM2				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	125	125		E0.H0.			45 30 0D 48 30 0D	
16	63	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
63	109	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
109	125	16	Z****.			5A 2A 2A 2A 2A 0D		
125	313	188	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
125	422	297		z.			7A 0D	
313	359	46	X0****. X0****.		.	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		00
359	406	47	X0****. X0****.		.	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		00
406	422	16	S****.		~	53 2A 2A 2A 2A 0D		7E
422	484	62	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
422	469	47			|			7C
422	531	109		s2202.			73 32 32 30 32 0D	
469	516	47			|			7C
484	531	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
516	563	47			|			7C
531	578	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
531	781	250		H0.E0.			48 30 0D 45 30 0D	

CAM1-CAM3				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	203	203		E0.H0.			45 30 0D 48 30 0D	
31	78	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
78	125	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
187	203	16	Z****.			5A 2A 2A 2A 2A 0D		
203	375	172	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
203	484	281		z.			7A 0D	
375	421	46	X0****. X0****.		|	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		7C
421	484	63	X0****. X0****.		|	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		7C
484	500	16	S****.		x |	53 2A 2A 2A 2A 0D		78 7C
500	546	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
500	843	343		s4104.			73 34 31 30 34 0D	
500	531	31			|			7C
531	593	62			|			7C
546	593	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
593	640	47			|			7C
593	640	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
640	687	47			|			7C
640	687	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
687	734	47			|			7C
687	750	63	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
750	796	46	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
796	843	47	I50505050394670..		|	49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		7C
843	890	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
843	1328	485		E0.H0.			45 30 0D 48 30 0D	

CAM1-CAM4				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	312	312		E0.H0.			45 30 0D 48 30 0D	
203	250	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
250	296	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
296	312	16	Z****.			5A 2A 2A 2A 2A 0D		
312	500	188	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
312	500	188		z.			7A 0D	
500	515	15		E0.H0.			45 30 0D 48 30 0D	
500	546	46	X0****. X0****.		.	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		00
531	609	78		E0.H0.			45 30 0D 48 30 0D	
546	593	47	X0****. X0****.		.	58 30 2A 2A 2A 2A 0D 58 30 2A 2A 2A 2A 0D		00
593	609	16			`			60
609	625	16	S****.			53 2A 2A 2A 2A 0D		
609	625	16		s5105.			73 34 31 30 34 0D	
609	703	94			`			60
625	671	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
671	718	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
703	765	62			`			60
718	765	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
765	812	47			`			60
765	812	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
812	859	47			`			60
812	859	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
859	906	47			`			60
906	953	47			`			60
906	968	62	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
953	1000	47			`			60
968	1015	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
1000	1046	46			`			60
1015	1062	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
1046	1109	63			`			60
1046	1531	485		E0.H0.			45 30 0D 48 30 0D	

Archivo Excel: ..\Ficheros anexos\tablas de mediciones\Tabla de Protocolo cambio de cámaras_ panel a Hub.xlsx

Pre-set

CAM1								
PRESET1				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	157	157		E0.H0.			45 30 0D 48 30 0D	
0	47	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
47	94	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
94	141	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
141	188	47	R0****.			52 30 2A 2A 2A 2A 0D		
157	500	343		s0100.			73 30 31 30 30 0D	
188	235	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
235	282	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
282	329	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
329	376	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
376	423	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
423	470	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
470	517	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
500	1000	500		E0.H0.			45 30 0D 48 30 0D	

CAM2								
PRESET1				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	125	125		H0.E0.			48 30 0D 45 30 0D	
15	62	47			|			7C
31	78	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
62	125	63			|			7C
78	125	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
125	145	20	R0****.			52 30 2A 2A 2A 2A 0D		
125	172	47			~			7E
125	265	140		s0100.			73 30 31 30 30 0D	
145	172	27	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
172	219	47			|			7C
172	219	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
219	265	46			|			7C
219	265	46	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
265	500	235		H0.E0.			48 30 0D 45 30 0D	

[Archivo Excel: ..\Ficheros anexos\tablas de mediciones\ Tabla de medidas_PresetCommand_panel a Hub.xlsx]

Comandos Pan/Tilt, Focus, Zoom e Iris

PAN_LEFT				
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_TX
0	47	47	I50505050394670&.	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 EF 0D
94	141	47	I50505050394670'.	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 EF 0D
141	188	47	I50505050394670'.	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 EF 0D
188	235	47	I53505050394670'.	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 F2 0D
235	282	47	I61505050394670'.	49 36 31 35 30 35 30 35 30 33 39 34 36 37 30 FA 0D
282	329	47	I80505050394670'.	49 38 30 35 30 35 30 35 30 33 39 34 36 37 30 09 0D
282	329	47	I88505050394670'.	49 38 38 35 30 35 30 35 30 33 39 34 36 37 30 12 0D
329	376	47	I92505050394670'.	49 39 32 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
376	423	47	I99505050394670'.	49 39 39 35 30 35 30 35 30 33 39 34 36 37 30 23 0D
423	470	47	I99505050394670'.	49 39 39 35 30 35 30 35 30 33 39 34 36 37 30 23 0D
470	517	47	I99505050394670'.	49 39 39 35 30 35 30 35 30 33 39 34 36 37 30 23 0D

PAN_RIGHT				
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_TX
0	47	47	I395050503946 70!.	49 33 39 35 30 35 30 35 30 33 39 34 36 37 30 3C 0D
94	141	47	I325050503946 70..	49 33 32 35 30 35 30 35 30 33 39 34 36 37 30 35 0D
141	188	47	I225050503946 70..	49 32 32 35 30 35 30 35 30 33 39 34 36 37 30 2A 0D
188	235	47	I155050503946 70..	49 31 35 35 30 35 30 35 30 33 39 34 36 37 30 24 0D
235	282	47	I105050503946 70..	49 31 30 35 30 35 30 35 30 33 39 34 36 37 30 1F 0D
282	329	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
282	329	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
329	376	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
376	423	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
423	470	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
470	517	47	I015050503946 70..	49 30 31 35 30 35 30 35 30 33 39 34 36 37 30 16 0D

ZOOM_TELE				
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_TX
0	47	47	I505054 50394670..	49 35 30 35 30 35 34 35 30 33 39 34 36 37 30 1E 0D
94	141	47	I505058 50394670".	49 35 30 35 30 35 38 35 30 33 39 34 36 37 30 22 0D
141	188	47	I505060 50394670..	49 35 30 35 30 36 30 35 30 33 39 34 36 37 30 1B 0D
188	235	47	I505064 50394670..	49 35 30 35 30 36 34 35 30 33 39 34 36 37 30 1F 0D
235	282	47	I505066 50394670!.	49 35 30 35 30 36 36 35 30 33 39 34 36 37 30 21 0D
282	329	47	I505075 50394670!.	49 35 30 35 30 37 35 35 30 33 39 34 36 37 30 21 0D
282	329	47	I505078 50394670$.	49 35 30 35 30 38 35 35 30 33 39 34 36 37 30 24 0D
329	376	47	I505081 50394670..	49 35 30 35 30 38 31 35 30 33 39 34 36 37 30 1E 0D
376	423	47	I505086 50394670#.	49 35 30 35 30 38 36 35 30 33 39 34 36 37 30 23 0D
423	470	47	I505088 50394670%.	49 35 30 35 30 38 38 35 30 33 39 34 36 37 30 1A 0D
470	517	47	I505099 50394670'.	49 35 30 35 30 39 39 35 30 33 39 34 36 37 30 27 0D

ZOOM_WIDE				
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_TX
0	47	47	I505045 50394670..	49 35 30 35 30 34 35 35 30 33 39 34 36 37 30 1E 0D
94	141	47	I505038 50394670 .	49 35 30 35 30 33 38 35 30 33 39 34 36 37 30 20 0D
141	188	47	I505032 50394670..	49 35 30 35 30 33 32 35 30 33 39 34 36 37 30 1A 0D
188	235	47	I505026 50394670..	49 35 30 35 30 32 36 35 30 33 39 34 36 37 30 1D 0D
235	282	47	I505022 50394670..	49 35 30 35 30 32 32 35 30 33 39 34 36 37 30 19 0D
282	329	47	I505020 50394670..	49 35 30 35 30 32 30 35 30 33 39 34 36 37 30 17 0D
282	329	47	I505015 50394670..	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1B 0D
329	376	47	I505008 50394670..	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1D 0D
376	423	47	I505001 50394670..	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
423	470	47	I505001 50394670..	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 16 0D
470	517	47	I505001 50394670..	49 35 30 35 30 35 30 35 30 33 39 34 36 37 30 16 0D

Archivo Excel: ..\Ficheros anexos\tablas de mediciones\ Tabla de medidas_PanTiltZoomFocusIris Commands_Panel a Hub.xlsx

Balance de blancos ATW:

CAM2								
ATW				ASCII		HEXA		
T1(ms)	T2(ms)	T(ms)	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY	BUS_P/T_TX	BUS_P/T_RX	BUS_TALLY
0	250	250		H0.E0.			48 30 0D 45 30 0D	
0	47	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
47	94	47			|			7C
47	94	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
94	141	47			|			7C
94	141	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
141	157	16	W0****.			57 30 2A 2A 2A 2A 0D		
141	188	47			~			7E
157	188	31	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
188	235	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
188	235	47			|			7C
235	282	47	I50505050394670..			49 35 30 35 30 35 30 35 30 35 30 33 39 34 36 37 30 1A 0D		
235	282	47			|			7C
250	500	250		H0.E0.			48 30 0D 45 30 0D	

Archivo Excel: ..\Ficheros anexos\tablas de mediciones\Tabla de medidas_ATW-BalanceDeBlancos_Panel a Hub.xlsx

Interruptores del panel de Condiciones (SW_Commands)

	BUS_P/T_TX	Byte de información de interruptores SW								
INTERRUPTORES	ASCII	bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]	Valor Hex
LAMP SW	I5050505039**1**670..	0	0	1	1	0	0	0	1	$31
IRIS SW	I5050505039**2**670..	0	0	1	1	0	0	1	0	$32
POWER SW	I5050505039**4**670..	0	0	1	1	0	1	0	0	$34
DEF SW	I505050503946**7**0..	0	0	1	1	0	1	1	0	$36
WIPE SW	I505050503945**7**0..	0	0	1	1	0	1	0	1	$35
HEAT SW	I505050503943**7**0..	0	0	1	1	0	0	1	1	$33
EXT SW	I505050503946**6**0..	0	0	1	1	0	1	1	0	$36
ND SW	I505050503946**5**0..	0	0	1	1	0	1	0	1	$35
S SW	I505050503946**3**0..	0	0	1	1	0	0	1	1	$33
SPEED SW_OFF	I505050503946**3**0..	0	0	1	1	0	0	0	0	$30
SPEED SW_ON	I505050503946**3**1..	0	0	1	1	0	0	0	1	$31

Archivo Excel: ..\Ficheros anexos\tablas de mediciones\Tabla de mediciones_SW Commands_Panel a Hub.xlsx

4. Protocolos de Comunicación de la cabeza Pan/Tilt (AW-PH300A/500/600)

4.1. Protocolo de comunicación RS422: Del Panel de control [AW- RP505] al Hub [AW-HB505]

4.1.1. Características generales:

Como hemos explicado con anterioridad en el apartado de "Especificaciones del interface PAN/TILT CONTROL RS-422", el interface está formado por 3 buses: Bus de Tx P/T, Bus de Rx P/T y Bus de Tally.

La comunicación entre el panel y el Hub es periódica, es decir, ambos se comunican continuamente repitiendo una secuencia de comandos.

El Hub por medio del "Bus Rx P/T" envía un comando de resultado "H0.E0.", que indica si se han producido errores y marca los periodos o ciclos de la comunicación entre ambos.

Este comando de resultado "H0.E0." está formado por dos comandos. Uno "H0." muestra la condición de la lámpara y el otro "E0." que no se ha producido ningún error en la comunicación en la cabeza P/T.

En el caso de que se produzca un error, este comando de resultado se interrumpe y el Hub envía uno comandos de mensaje de error: "E****." (si el error se produce en el panel) y "E." (si el error se ha producido en la cabeza P/T).

Como hemos observado en las mediciones realizadas, para las cámaras 1,3 y 4 el periodo de envió de este comando es cada 500ms. En cambio para la cámara 2 el periodo es cada 250ms.

Dentro de este periodo (de 250ms o 500ms), el panel le envía al hub una secuencia de comandos cada [47-60]ms, formada por repeticiones: Del "StatusCommand "a través del "Bus Rx P/T" y de un comando de Tally, pero este último se envía por medio del "Bus Tally".

Ambos comandos son enviados a la vez, de esta forma el Hub sabe que cabeza de cámara P/T está siendo controlada por el panel.

En este hipervínculo hay un ejemplo animado de la secuencia de comunicación para la Cámara 3 en un ciclo:

Ver Clip: ..\Ficheros anexos\animaciones Flash\Animacion_StatusCommand.html

En las siguientes tablas se muestra las secuencias para cada cámara, en único ciclo:

Para la cámara 2 el periodo del ciclo es de 250ms:

CAM2			ASCII		
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	BUS_TALLY
0	250	250		H0.E0.	
0	47	47	I50505050504770..		|
47	94	47	I50505050504770..		|
94	141	47	I50505050504770..		|
141	188	47	I50505050504770..		|
188	235	47	I50505050504770..		|
235	282	47	I50505050504770..		|
250	500	250		H0.E0.	

CAM3			ASCII		
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	BUS_TALLY
0	500	500		E0.H0.	
0	47	47	I50505050394670..		|
47	94	47	I50505050394670..		|
94	141	47	I50505050394670..		|
141	188	47	I50505050394670..		|
188	235	47	I50505050394670..		|
235	282	47	I50505050394670..		|
282	329	47	I50505050394670..		|
329	376	47	I50505050394670..		|
376	423	47	I50505050394670..		|
423	470	47	I50505050394670..		|
470	517	47	I50505050394670..		|
500	1000	500		E0.H0.	

CAM4			ASCII		
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	BUS_TALLY
0	500	500		E0.H0.	
0	47	47	I50505050394670..		60
47	94	47	I50505050394670..		60
94	141	47	I50505050394670..		60
141	188	47	I50505050394670..		60
188	235	47	I50505050394670..		60
235	282	47	I50505050394670..		60
282	329	47	I50505050394670..		60
329	376	47	I50505050394670..		60
376	423	47	I50505050394670..		60
423	470	47	I50505050394670..		60
470	517	47	I50505050394670..		60
500	1000	500		E0.H0.	

Para la cámara 1 el comando de tally no se envía, en cambio para el resto de cámaras sí.

CAM1			ASCII		
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	BUS_TALLY
0	500	500		E0.H0.	
0	47	47	I50505050394670..		
47	94	47	I50505050394670..		
94	141	47	I50505050394670..		
141	188	47	I50505050394670..		
188	235	47	I50505050394670..		
235	282	47	I50505050394670..		
282	329	47	I50505050394670..		
329	376	47	I50505050394670..		
376	423	47	I50505050394670..		
423	470	47	I50505050394670..		
470	517	47	I50505050394670..		
500	1000	500		E0.H0.	

Cambio de cámara

El cambio de cámara se produce a través de una interrupción formada por una secuencia de comandos, que indican al Hub la cámara seleccionada por el panel. Y este responde que el cambio ha sido establecido, enviando el valor del último Pre-set seleccionado “s2202.”por dicha cámara.

Esta secuencia de comandos está formada por 4 etapas:

1. Corte de la línea: Cortamos la línea (Z****.) para dejarla libre e introducir el cambio de cámara.

   El Hub devuelve el resultado (z.), que le indica al panel que la línea está cortada y ya puede enviar los comandos de la cámara seleccionada.

2. Selección de cámara: El panel envía dos veces el comando (X0****. X0****.) acompañado de comando de tally, que indica qué cámara ha sido seleccionada.

   El comando de Tally para la selección de cámara está formado por dos valores numéricos de un digito que indican la cámara seleccionada.

	Comando Tally	
Cámara	Valor ASCII	Valor HEX
1	NONE	NONE
2	00 ~	00 7E
3	x |	78 7C
4	. .	00 00

3. Petición de información del último Pre-set establecido en esa misma cámara (S****.).

4. Establece el ciclo normal de la secuencia StatusCommand para esa cámara. Enviando un nuevo valor del comando de Tally constante:

	Comando Tally	
Cámara	Valor ASCII	Valor HEX
1	NONE	NONE
2	|	7C
3	|	7C
4	`	60

En este hipervínculo hay un ejemplo animado de la secuencia de cambio de cámara para la Cámara 3 en un ciclo:

Ver Clip: ..\Ficheros anexos\animaciones Flash\Animacion_CambioDeCamara.html

En las siguientes tablas mostramos las secuencias de comandos para el cambio de cámaras en un ciclo y de la cámara 1 a la cámara seleccionada:

CAM1-CAM2			BUS P/T		BUS_TALLY
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	CAM 1 a 2
0	125	125		E0.H0.	
16	63	47	I50505050394670..		
63	110	47	I50505050394670..		
110	126	16	Z****.		
126	314	188	I50505050394670..		
126	423	297		z.	
314	361	47	X0****. X0****.		.
361	407	47	X0****. X0****.		.
407	423	16	S****.		~
423	470	47	I50505050394670..		|
423	532	110		s2202.	
470	516	47	I50505050394670		|
516	563	47	I50505050394670..		|
532	781	250		H0.E0.	

CAM1-CAM3/CAM4			BUS P/T		BUS_TALLY	
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	CAM 1 a 3	CAM 1 a 4
0	203	203		E0.H0.		
0	47	47	I50505050394670..			
47	94	47	I50505050394670..			
94	110	16	Z****.			
110	298	188	I50505050394670..			
203	500	297		z.		
298	345	47	X0****. X0****.		|	.
345	392	47	X0****. X0****.		|	.
392	408	16	S****.		x	`
408	455	47	I50505050394670..		|	`
455	500	47	I50505050394670..		|	`
500	831	331		s0100.		
502	549	47	I50505050394670..		|	`
549	596	47	I50505050394670..		|	`
596	643	47	I50505050394670..		|	`
643	690	47	I50505050394670..		|	`
690	737	47	I50505050394670..		|	`
737	784	47	I50505050394670..		|	`
784	831	47	I50505050394670..		|	`
831	1331	500		E0.H0.		

CAM2-CAM1			BUS P/T		BUS_TALLY
T1(ms)	T2(ms)	T(ms)	BUS_TX_P/T	BUS_RX_P/T	CAM 2 a 1
0	188	188		E0.H0.	
31	78	47	I50505050394670..		|
78	125	47	I50505050394670..		|
125	172	47	I50505050394670..		|
172	188	16	Z****.		
188	376	188		z.	
188	391	203	I50505050394670..		
391	438	47	X0****. X0****.		
438	485	47	X0****. X0****.		
485	501	16	S****.		
376	547	171		s5105.	
501	548	47	I50505050394670..		
548	595	47	I50505050394670..		
547	1047	500		E0.H0.	

Control de Pre-set y balance de blancos (ATW)

Estos comandos son ejecutados como una interrupción dentro del ciclo normal de comunicación de Command Status. También van acompañados de forma sincronizada por el comando Tally con los siguientes valores:

	Comando Tally	
Cámara	Valor ASCII	Valor HEX
1	NONE	NONE
2	~	7E
3	|	7C
4	p	70

En la siguiente tabla se muestra un ejemplo de cómo se envían estos comandos para cada cámara.

Preset 0	BUS P/T		Bus TALLY			
T(ms)	BUS_TX_P/T	BUS_RX_P/T	Cam 1	Cam 2	Cam 3	Cam 4
47	I50505050394670..		NONE	|	|	`
47	R0****.		NONE	~	|	p
343		s0100.	NONE			
47	I50505050394670..		NONE	|	|	`

Así será para todos los comandos de control de Pre-set (Seleccionar/Guardar), los distintos balances de blancos (ATW, Canal A y Canal B) y el comando de establecer un límite de posición P/T, lo único que variará será el comando de operación enviado por el correspondiente para cada función.

4.1.2. Comando para el Joystick y el panel de información de interruptores (Status Command)

Es un comando que se ejecuta de forma continua y establece el estado actual de los joysticks de rotación P/T, de las lentes (Zoom, Focus e Iris) y del estado del panel de los interruptores (SW).

El comando se debe enviar como máximo cada 60msec. Si el ciclo de acción es mayor de 60msec, el movimiento de la Pan / Tilt se convierte entorpece.

Sintaxis → **(Iaabbccddeefghijk)**

Ejemplo→ Para I50505050504770..→en Hexa. es → 49 35 30 35 30 35 30 35 30 35 30 34 37 37 CB 0D

I : Identificador de datos ID de información -> establece el operador 'I' del Status Command.
aa : Valor numérico de dos dígitos que indica la dirección y la velocidad del movimiento rotatorio Pan. Cuyo rango está comprendido entre 00..99:

Velocidad Máxima hacia la izquierda -> '00'
Velocidad Mínima hacia la izquierda -> '49'
Parada (Stop) -> '50'
Velocidad Mínima hacia la derecha -> '51'
Velocidad Máxima hacia la derecha -> '99'

bb : Valor numérico de dos dígitos que indica la dirección y la velocidad del movimiento rotatorio Tilt. Cuyo rango está comprendido entre 00..99:

Velocidad Máxima para la dirección abajo -> '00'
Velocidad Mínima para la dirección abajo -> '49'
Stop -> '50'
Velocidad Mínima para la dirección arriba -> '51'
Velocidad Máxima para la dirección arriba -> '99'

cc : Valor numérico de dos dígitos que indica el desplazamiento y la velocidad del Zoom. Cuyo rango está comprendido entre 00…99.

Velocidad Máxima para WIDE -> '00'
Velocidad Mínima para WIDE -> '49'
Stop -> '50'
Velocidad Mínima para TELE -> '51'
Velocidad Máxima para TELE -> '99'

dd : Valor numérico de dos dígitos que indica desplazamiento y la velocidad del Foco. Cuyo rango está comprendido entre 01..99.

Velocidad Máxima para NEAR -> '01'
Velocidad Mínima para NEAR -> '49'
Stop -> '50'
Velocidad Mínima para FAR -> '51'
Velocidad Máxima para FAR -> '99'

ee : : Valor numérico de dos dígitos que indica el nivel de Iris.

Rango del Nivel de iris-> de 1..99
Valor Mínimo CLOSE -> '01'
Valor Máximo OPEN -> '99'

f : Valor numérico de un byte de información (en Hexadecimal)
bit[0] : interruptor LAMP SW (ON:1 , OFF:0)
bit[1] : interruptor IRIS SW (ON:1 , OFF:0)
bit[2] : interruptor POWER SW (ON:1 , OFF:0)

	Byte de información de interruptores SW								
	bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]	Valor Hex
LAMP SW	0	0	1	1	0	0	0	1	$31
IRIS SW	0	0	1	1	0	0	1	0	$32
POWER SW	0	0	1	1	0	1	0	0	$34

g : Valor numérico de un byte de información (en Hexadecimal)
bit[0] : interruptor DEF SW (ON:0 , OFF:1)
bit[1] : interruptor WIPE SW (ON:0 , OFF:1)
bit[2] : interruptor HEAT SW (ON:0 , OFF:1)

	Byte de información de interruptores SW								
	bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]	Valor Hex
DEF SW	0	0	1	1	0	1	1	0	$36
WIPE SW	0	0	1	1	0	1	0	1	$35
HEAT SW	0	0	1	1	0	0	1	1	$33

h : numérico de un byte de información (en Hexadecimal)
bit[0] : interruptor EXT SW (ON:0 , OFF:1)
bit[1] : interruptor ND SW (ON:0 , OFF:1)
bit[2] : interruptor S SW (ON:0 , OFF:1)

	Byte de información de interruptores SW								
	bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]	Valor Hex
EXT SW	0	0	1	1	0	1	1	0	$36
ND SW	0	0	1	1	0	1	0	1	$35
S SW	0	0	1	1	0	0	1	1	$33

i : numérico de un byte de información (en Hexadecimal)
bit[0] : interruptor SPEED SW (Pulsado :1 , Normal :0)

	Byte de información de interruptores SW								
SPEED	bit[7]	bit[6]	bit[5]	bit[4]	bit[3]	bit[2]	bit[1]	bit[0]	Valor Hex
Normal	0	0	1	1	0	0	0	0	$30
Pulsado	0	0	1	1	0	0	0	1	$31

j : Check Sum -> suma de verificación (es la suma de longitud=1 byte, de los 14 bytes de información)

ChkS= HEX(a+a+ b+b+c+c+d+d+e+e+f+g+h+i) [Byte]

k : Delimitador (Current-CR)

4.1.3. Comando para el encendido (Power ON/OFF)

Comando para el apagado y encendido del Hub de Cámaras y las cabezas P/T conectadas al sistema.

P : Identificador de datos ID de información -> establece el operador 'P' del comando Power.
a : Valor de power on/off (ON : 1 , OFF : 0)
b : Dummy (bit line , 0b10101010)
c : Delimitador(Current-CR)

Sintaxis → **(Pabc)**

Ejemplo→ Para P1****.→en Hexa. es →50 31 2A 2A 2A 2A 0D

4.1.4. Resultado del comando Power ON/OFF

Resultado del comando Power ON/OFF mostrando el estado desactivación del Hub de cámaras.

p : Identificador de datos ID de información -> establece el operador 'p' de la respuesta Power.
a : Condición de Power on/off (ON : 1 , OFF : 0)
b : Delimitador (Current – CR)

Sintaxis → **(Pab)**

Ejemplo→ Para P1.→en Hexa. es →50 31 0D

4.1.5. Comando para el cambio de velocidad (Speed High/Low)

Comando para el cambio de velocidad o sensibilidad del joystick Pan/Tilt de alta a baja o viceversa, según su estado anterior.

r : Identificador de datos ID de información -> establece el operador 'r' del comando Speed High/Low.
a : Dummy (bit line, 0b10101010)
b : Delimitador (Current-CR)

Sintaxis → **(rab)**

Ejemplo→ Para **r****.→en Hexa. es →72 2A 2A 2A 2A 0D

4.1.6. Condición de la lámpara (Lamp ON/OFF)

Respuesta de la cabeza Pan/Tilt de la condición de la lámpara conectada a la cabeza P/T.

H : Identificador de datos ID de información -> establece el operador 'H' de la respuesta de la condición de LAMP.
a : Valor de la condición de la Lámpara (OFF : 0 , ON : 1, Detect for break :2)
b : Delimitador(Current-CR)

Sintaxis → **(Hab)**

Ejemplo→ Para **H0.**→en Hexa. es →48 30 0D

4.1.7. Comando para obtener la información de Pre-set memory

Este comando se envía cuando se encienda el sistema y al recuperar la comunicación.

S : Identificador de datos ID de información -> establece el operador 'S' del comando para obtener la información del Pre-set.
a : Dummy (bits line 0b10101010)
b : Delimitador(Current-CR)

Sintaxis → **(Sab)**

Ejemplo→ Para **S****.→en Hexa. es →53 2A 2A 2A 2A 0D

4.1.8. Comando para seleccionar Pre-set memory

Comando que selecciona y ejecuta una memoria de Pre-set.

R : Identificador de datos ID de información -> establece el operador 'R' del comando para seleccionar Pre-set memory
a : número de Pre-set (0..9)
b : Dummy (bits line 0b10101010)
c : Delimitador (Current-CR)

Sintaxis → **(Rabc)**

Ejemplo→ Para **R0****.→en Hexa. es →52 2A 2A 2A 2A 0D

4.1.9. Comando para guardar una posición en el Pre-set memory

Cuando ejecutamos este comando se guarda en la memoria de Pre-set seleccionada, el valor actual de la posición de rotación Pan/Tilt y las posiciones de las lentes.

M : Identificador de datos ID de información -> establece el operador 'M' del comando para una posición en el Pre-set

a : número de Pre-set (0..9)

b : Dummy (bits line 0b10101010)

c : Delimitador(Current-CR)

Sintaxis → **(Mabc)**

Ejemplo→ Para **M0****.**→en Hexa. es →4D 2A 2A 2A 2A 0D

4.1.10. Resultado del comando para Pre-set memory

Resultado procedente de la cabeza P/T al ejecutar el comando para seleccionar Pre-set memory, indica que Pre-set memory ha sido ejecutado.

s : Identificador de datos ID de información -> establece el operador 's' del resultado Pre-set memory

a : número de Pre-set (último número) (0..9)

b: Byte= ' 1 '

c: Byte= ' 0 '

d: número de Pre-set (último número seleccionado) (0..9)

e : Delimitador(Current-CR)

Sintaxis → **(sabcde)**

Ejemplo→ Para **s4104.**→en Hexa. es → 73 34 31 30 34 0D

4.1.11. Comando para establecer el balance de blancos (White balance)

Comando para establecer uno de los tres modos de balances de blancos: ATW, canal A y canal B.

W : Identificador de datos ID de información -> establece el operador 'W' del comando White Balance.

a : Modo de White balance.

0 : ATW

1 : ATW_Canal A

2 : ATW_Canal B

b : Dummy(bits line 0b10101010)

c : Delimitador(Current-CR)

Sintaxis → **(Wabc)**

Ejemplo→ Para **W0****.**→en Hexa. es → 57 30 2A 2A 2A 2A 0D

4.1.12. Mensaje de error

Mensaje de error de comunicación, que se produce cuando hay un fallo en el Panel.

E : Identificador de datos ID de información -> establece el operador 'E' de la repuesta de un error.
a : Dummy (bits line : 0b10101010)
b : Delimitador(Current-CR)

Sintaxis → **(Eab)**

Ejemplo→ Para **E****.→**en Hexa. es →** 45 2A 2A 2A 2A 0D

4.1.13. Mensaje de error Pan/Tilt

Mensaje de error de comunicación, que se produce cuando hay un fallo en la cabeza P/T.

E : Identificador de datos ID de información -> establece el operador 'E' de la repuesta de un error.
a : Delimitador(Current-CR)

Sintaxis → **(Ea)**

Ejemplo→ Para **E.→**en Hexa. es →** 45 0D

4.1.14. Comando para establecer/Deshabilitar la posición limite Pan/Tilt (LIMIT)

Comando para el establecer o resetear los límites de rotación de la cabeza P/T.

L : Identificador de datos ID de información -> establece el operador 'L' del comando LIMIT.
a : Valor numérico del 1 al 4 que indica cual es el límite establecido:
1 : limit for upper side (TILTE)
2 : limit for lower side (TILTE)
3 : limit for left side(PAN)
4 : limit for right side(PAN)
b : dummy (bits line : 0b10101010)
c : Delimitador (Current-CR)

Sintaxis → **(Labc)**

Ejemplo→ Para **L1****.→**en Hexa. es →** 4C 31 2A 2A 2A 2A 0D

4.1.15. Resultados del comando para Establecer/ Resetear el límite de posición (LIMIT)

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando Limit nos devuelve el resultado del estado del límite: Activado/Desactivado.

l : Identificador de datos ID de información -> Identificador de la respuesta del comando Limit 'l'.
a : Valor numérico que muestra la condición del límite (0->Reset, 1->Setting).
b : Delimitador (Current-CR)

Sintaxis → **(lab)**

Ejemplo→ Para **L0.→**en Hexa. es →** 6C 30 0D

4.1.16. Comando para el corte de línea (CUT OFF LINE)

Comando para producir un corte en la línea de comunicación.

Z : Identificador de datos ID de información -> Identificador del comando Cut Off Line.
a : Dummy (bits line : 0b10101010)
b : Delimitador(Current-CR)

Sintaxis → **(Zab)**

Ejemplo→ Para **Z****** .→en Hexa. es → 5A 2A 2A 2A 2A 0D

4.1.17. Resultados del comando para el corte de línea

En el caso de que la ejecutemos un corte de línea, este comando será enviado.

z : Identificador de datos ID de información -> Identificador de la respuesta del comando Cut Off Line
a : Delimitador(Current-CR)

Sintaxis → **(Za)**

Ejemplo→ Para **Z.**→en Hexa. es → 5A 0D

4.2. Protocolo de comunicación RS232C: Del PC a la cabeza Pan/Tilt [AW-PH300A/500/600]

4.2.1. Características generales:

Los comandos que se envían a la cabeza P/T (AW-PH300A/500/600) desde el PC no se emiten periódicamente, sino se tratan como eventos. Es decir, se envían o se reciben únicamente una vez y la cabeza P/T ejecuta el comando hasta que recibe otro.

El intervalo de tiempo mínimo de envío de comandos debe realizarse entre [45-60] ms, para que no se produzca errores como: movimientos torpes de la cabeza P/T causados por la interferencia del envío constante de comandos en un periodo inferior a 30ms.

Podemos controlar hasta un máximo 50 Pre-Sets para todos los modelos de cabeza P/T. Cosa que en el protocolo de comunicación RS-422 solo se puede controlar un máximo 10 Pre-Sets.

Lo único que no se puede controlar es el balance de blancos ATW, mientras que en el protocolo de comunicación RS-422 sí que lo permite.

También hay comandos que nos permiten establecer o recibir la posición de rotación Pan/Tilt y de las lentes. Esta posición estará formada por un número de cuatro dígitos en Hexadecimal para Pan y el otro para Tilt. Y un número de tres dígitos para cada una de las posiciones del Zoom, Focus e Iris. Esta última característica únicamente está disponible para el modelo AW-PH360L.

En la siguiente tabla se muestra el rango en grados de los valores de posición Pan/Tilt y los porcentajes de la posición Focus, Zoom e Iris. Con sus valores correspondientes en decimal y en hexadecimal.

	RANGO	Valor [Hex.]	Valor [Decimal]
PAN	0º..300º	0000..FFFF	0..65535
TILT	0º..190º	0000..FFFF	0..65535
ZOOM	0..99	000..FFF	0..4095
FOCUS	0..99	000..FFF	0..4095
IRIS	0..99	000..FFF	0..4095

Sabiendo esto, calcularemos cuales el desplazamiento para un 1º de los comandos Pan/Tilt:

- Si el máximo valor de posición Pan es un ángulo de Θ(300º)=$\mathrm{FFFF}_{(Hex)}$ =65535

$$\mathrm{Despl\_Pan}_{\min}(\text{para } 1º) = \frac{65535}{300º} \approx 218 = \mathrm{DA}_{(\mathrm{Hex})}$$

- Si el máximo valor de posición Tilt es un ángulo de Θ(190º)= $\mathrm{FFFF}_{(Hex)}$=65535

$$\mathrm{Despla\_Tilt}_{\min}(\text{para } 1º) = \frac{65535}{190º} \approx 344 = 158_{(\mathrm{Hex})}$$

Ahora realizaremos lo mismo para los comandos de posición de las lentes para un desplazamiento de un 1%:

- Si el máximo valor de posición es %(99)= $\mathrm{FFF}_{(Hex)}$=4095

$$\mathrm{Despl\_Lentes}_{min}(\text{para } 1\%) = \frac{4095}{99} \approx 41 = 29_{(Hex)}$$

Otra cosa a tener en cuenta es cuando estén conectados el PC y el panel de control, y ambos envíen un comando en el mismo instante, tendrá prioridad la operación de control iniciada desde el panel (AW-RP505).

Todos los comandos del protocolo de comunicación, los he programado para 'Visual Studio C++' (plataforma 'Microsoft .NET Framework') y los he encapsulado en un encabezado llamado: 'CamCommands.h'.

4.2.2. Comando para la operación panorámica PAN:

Comando para la ejecución del movimiento de rotación panorámico Pan.

Cuando ejecutemos el comando con un valor determinado entre 0..49 (direc. Izq.) ó entre 51..99 (direc. Der.), la cabeza P/T girará hasta que enviemos el comando de parada 'Stop'(#P50.) o hasta que llegue a la posición limite '0º ó 300º'.

Sintaxis → (**#Paab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
P : Identificador de datos ID de información -> establece el operador 'P' del comando Pan.
a a : Valor numérico de dos dígitos que indica la dirección y la velocidad del movimiento rotatorio Pan. Cuyo rango está comprendido entre 01..99:

Velocidad Máxima hacia la izquierda -> '01'
Velocidad Mínima hacia la izquierda -> '49'
Parada (Stop)-> '50'
Velocidad Mínima hacia la derecha -> '51'
Velocidad Máxima hacia la derecha -> '99'

b : Delimitador (Current-CR)

Ejemplo: Para '#P50.', en Hexa. es → '23 50 35 30 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_PanOperation(System::IO::Ports::SerialPort^
serialPort_BUS_CAM,int ValorDireccion){
 /*Comando para la operación del Pan.
      Valor de Dirección-> de 1..99
            Velocidad Máxima para la dirección izquierda -> '01'
            Velocidad Mínima para la dirección izquierda -> '49'
            Stop -> '50'
            Velocidad Mínima para la dirección derecha -> '51'
            Velocidad Máxima para la dirección derecha -> '99' */

      //Variable de la operador Pan -> #P50.
       array<unsigned char>^ bufferSalidaPan=gcnew array<unsigned char>(5);

      //valor inicial del operador Pan -> #P50. (ASCII)
       bufferSalidaPan[0]=0x23; //Valor que indica comamndos del Pc -> '#'
       bufferSalidaPan[1]=0x50; //Valor de información ID -> 'P'
       bufferSalidaPan[2]=0x35; //Valor Pan joystick -> Bytes 'aa'
       bufferSalidaPan[3]=0x30;
       bufferSalidaPan[4]=0x0D; //Valor Delimiter (Current CR)

       //Convertimos el valor de dirección en decenas y unidades
      int ValorDireccion_decenas=0;
      int ValorDireccion_unidades=0;
      ValorDireccion_decenas=ValorDireccion/10;
      ValorDireccion_unidades=ValorDireccion-ValorDireccion_decenas*10;
```

```
        //valor de MCP pan de cámara -> #Paa.
        //aa-> numero de dos dígitos del ValorDireccion
        bufferSalidaPan[2]=0x30+(unsigned char)abs(ValorDireccion_decenas);
        bufferSalidaPan[3]=0x30+(unsigned char)abs(ValorDireccion_unidades);

        //envió de valor de Pan
         serialPort_BUS_CAM->Write(bufferSalidaPan,0,5);
 }//Evento_PanOperation()
```

4.2.3. Comando para la operación de inclinación Tilt

Comando para la ejecución del movimiento de rotación de inclinación Tilt.

Cuando ejecutemos el comando con un valor determinado entre 0..49 (direc. hacia Abajo) ó entre 51..99 (direc. hacia Arriba), la cabeza P/T girará hasta que enviemos el comando de parada 'Stop'(#T50.) o hasta que llegue a la posición limite '0º ó 195º'.

Sintaxis → (**#Taab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
T : Identificador de datos ID de información -> establece el operador 'T' del comando Tilt.
a a : Valor numérico de dos dígitos que indica la dirección y la velocidad del movimiento rotatorio Tilt. Cuyo rango está comprendido entre 01..99:

Velocidad Máxima para la dirección abajo -> '01'
Velocidad Mínima para la dirección abajo -> '49'
Stop -> '50'
Velocidad Mínima para la dirección arriba -> '51'
Velocidad Máxima para la dirección arriba -> '99'

b : Delimitador (Current-CR)

Ejemplo: Para '#T50.', en Hexa. es → '23 54 35 30 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_TiltOperation(System::IO::Ports::SerialPort^
serialPort_BUS_CAM,int ValorDireccion){
       /*Comando para la operación del Tilt
       Valor de Dirección-> de 1..99
            Velocidad Máxima para la dirección abajo -> '01'
            Velocidad Mínima para la dirección abajo -> '49'
            Stop -> '50'
            Velocidad Mínima para la dirección arriba -> '51'
            Velocidad Máxima para la dirección arriba -> '99' */

      //Variable de la operador Tilt -> #T50.
      array<unsigned char>^bufferSalidaTilt=gcnew array<unsigned char>(5);

      //valor inicial del operador Tilt -> #T50. (ASCII)
       bufferSalidaTilt[0]=0x23; //Valor que indica comamndos del Pc
       bufferSalidaTilt[1]=0x54; //Valor de información ID
       bufferSalidaTilt[2]=0x35; //Valor Tilt joystick -> Bytes 'aa'
       bufferSalidaTilt[3]=0x30;
       bufferSalidaTilt[4]=0x0D; //Valor Delimiter (Current CR)

       //Convertimos el valor dirección a unidades y decenas.
      int ValorDireccion_decenas=0;
      int ValorDireccion_unidades=0;

      ValorDireccion_decenas=ValorDireccion/10;
      ValorDireccion_unidades=ValorDireccion-ValorDireccion_decenas*10;
```

```
      //Introducimos el valor de dirección en el buffer de salida.
      //valor de MCP Tilt de cámara -> #Taa.
      //aa-> numero de dos dígitos del ValorDireccion
      bufferSalidaTilt[2]=0x30+(unsigned char)abs(ValorDireccion_decenas);
      bufferSalidaTilt[3]=0x30+(unsigned char)abs(ValorDireccion_unidades);


      //envió de valor de Tilt
       serialPort_BUS_CAM->Write(bufferSalidaTilt,0,5);
 }//Evento_TiltOperation()
```

4.2.4. Comando para la operación del Zoom

Comando para la ejecución del movimiento del Zoom.

Cuando ejecutemos el comando con un valor determinado entre 0..49 (direc. hacia Gran Angular 'WIDE') ó entre 51..99 (direc. hacia Tele Objetivo 'TELE'), la lente del Zoom se moverá hasta que enviemos el comando de parada 'Stop'(#Z50.) o hasta que llegue a la posición limite '1 ó 99'.

Sintaxis → (**#Zaab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
Z : Identificador de datos ID de información -> establece el operador 'Z' del comando Zoom.
a a : Valor numérico de dos dígitos que indica el desplazamiento y la velocidad del Zoom. Cuyo rango está comprendido entre 01…99.

Velocidad Máxima para WIDE -> '01'
Velocidad Mínima para WIDE -> '49'
Stop -> '50'
Velocidad Mínima para TELE -> '51'
Velocidad Máxima para TELE -> '99'

b : Delimitador (Current-CR)

Ejemplo: Para '#Z50.', en Hexa. es → '23 5A 35 30 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_ZoomOperation(System::IO::Ports::SerialPort^
serialPort_BUS_CAM,int ValorDireccion){
 /*Comando para la operación del Zoom
 Valor de Dirección-> de 1..99
      Velocidad Máxima para WIDE -> '01'
      Velocidad Mínima para WIDE -> '49'
      Stop -> '50'
      Velocidad Mínima para TELE -> '51'
      Velocidad Máxima para TELE -> '99'   */

       //Variable de la operador Zoom -> #Z50.
      array<unsigned char>^bufferSalidaZoom=gcnew array<unsigned char>(5);

      //valor inicial del operador Zoom -> #Z50. (ASCII)
       bufferSalidaZoom[0]=0x23; //Valor que indica comamndos del Pc
       bufferSalidaZoom[1]=0x5A; //Valor de información ID
       bufferSalidaZoom[2]=0x35; //Valor Zoom joystick -> Bytes 'aa'
      bufferSalidaZoom[3]=0x30;
       bufferSalidaZoom[4]=0x0D; //Valor Delimiter (Current CR)
```

```
        //Convertimos el valor dirección a unidades y decenas.
       int ValorDireccion_decenas=0;
       int ValorDireccion_unidades=0;

       ValorDireccion_decenas=ValorDireccion/10;
       ValorDireccion_unidades=ValorDireccion-ValorDireccion_decenas*10;

       //Introducimos el valor de dirección en el buffer de salida.
       //valor de MCP Zoom de cámara -> #Zaa.
       //aa-> numero de dos dígitos del ValorDireccion
       bufferSalidaZoom[2]=0x30+(unsigned char)abs(ValorDireccion_decenas);
       bufferSalidaZoom[3]=0x30+(unsigned char)abs(ValorDireccion_unidades);


       //envió de valor de Zoom
        serialPort_BUS_CAM->Write(bufferSalidaZoom,0,5);

 }//Evento_ZoomOperation()
```

4.2.5. Comando para la operación del Foco

Comando para la ejecución del movimiento del enfoque 'FOCUS'.

Cuando ejecutemos el comando con un valor determinado entre 0..49 (direc. Acercar 'NEAR') ó entre 51..99 (direc. Alejar 'FAR'), la lente del Zoom se moverá hasta que enviemos el comando de parada 'Stop'(#F50.) o hasta que llegue a la posición limite '1 ó 99'.

Sintaxis → (**#Faab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
F : Identificador de datos ID de información -> establece el operador 'F' del comando Foco.
a a : Valor numérico de dos dígitos que indica desplazamiento y la velocidad del Foco. Cuyo rango está comprendido entre 01..99.

Velocidad Máxima para NEAR -> '01'
Velocidad Mínima para NEAR -> '49'
Stop -> '50'
Velocidad Mínima para FAR -> '51'
Velocidad Máxima para FAR -> '99'

b : Delimitador (Current-CR)

Ejemplo: Para '#F50.', en Hexa. es → '23 46 35 30 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_FocusOperation(System::IO::Ports::SerialPort^
serialPort_BUS_CAM, int ValorDireccion){
 /*Comando para la operación del Zoom
      Valor de Dirección-> de 1..99
      Velocidad Máxima para NEAR -> '01'
      Velocidad Mínima para NEAR -> '49'
      Stop -> '50'
      Velocidad Mínima para FAR -> '51'
      Velocidad Máxima para FAR -> '99' */

      //Variable de la operador Focus -> #F50.
      array<unsigned char>^bufferSalidaFocus=gcnew array<unsigned char>(5);
```

```
      //valor inicial del operador Focus -> #F50. (ASCII)
       bufferSalidaFocus[0]=0x23; //Valor que indica comamndos del Pc
       bufferSalidaFocus[1]=0x46; //Valor de información ID
       bufferSalidaFocus[2]=0x35; //Valor Focus joystick -> Bytes 'aa'
       bufferSalidaFocus[3]=0x30;
       bufferSalidaFocus[4]=0x0D; //Valor Delimiter (Current CR)

       //Convertimos el valor dirección a unidades y decenas.
      int ValorDireccion_decenas=0;
      int ValorDireccion_unidades=0;
      ValorDireccion_decenas=ValorDireccion/10;
      ValorDireccion_unidades=ValorDireccion-ValorDireccion_decenas*10;

      //Introducimos el valor de dirección en el buffer de salida.
      //valor de MCP Focus de cámara -> #Faa.
      //aa-> numero de dos dígitos del ValorDireccion
      bufferSalidaFocus[2]=0x30+(unsigned char)abs(ValorDireccion_decenas);
      bufferSalidaFocus[3]=0x30+(unsigned char)abs(ValorDireccion_unidades);


      //envió de valor de Focus
       serialPort_BUS_CAM->Write(bufferSalidaFocus,0,5);
}//Evento_FocusOperation()
```

4.2.6. Comando para la operación del IRIS

Comando para establecer el nivel del Iris 'IRIS'. El rango del nivel del Iris está comprendido entre 01..99.

Sintaxis → (**#Iaab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
I : Identificador de datos ID de información -> establece el operador 'I' del comando IRIS.
a a : Valor numérico de dos dígitos que indica el nivel de Iris.
Rango del Nivel de iris-> de 1..99
Valor Mínimo CLOSE -> '01'
Valor Máximo OPEN -> '99'
b : Delimitador (Current-CR)

Ejemplo: Para '#I50.', en Hexa. es → '23 49 35 30 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_IrisOperation (System::IO::Ports::SerialPort^
serialPort_BUS_CAM, int NivelIris){
 /*Comando para la operación del IRIS
      Valor del Nivel de iris-> de 1..99
      Valor Máximo CLOSE -> '01'
      Valor Mínimo OPEN -> '99' */

       //Variable de la operador Iris -> #I50.
       array<unsigned char>^bufferSalidaIris=gcnew array<unsigned char>(5);

      //valor inicial del operador Iris -> #I50. (ASCII)
       bufferSalidaIris[0]=0x23; //Valor que indica comamndos del Pc
       bufferSalidaIris[1]=0x49; //Valor de información ID
       bufferSalidaIris[2]=0x35; //Valor Iris joystick -> Bytes 'aa'
       bufferSalidaIris[3]=0x30;
       bufferSalidaIris[4]=0x0D; //Valor Delimiter (Current CR)
```

```
        //Convertimos el valor dirección a unidades y decenas.
        int DEC_decimal=0;
        int DEC_unidad=0;
        DEC_decimal=NivelIris/10;
        DEC_unidad=NivelIris-DEC_decimal*10;

        //Introducimos el valor de dirección en el buffer de salida.
        //valor de MCP Iris de cámara -> #Iaa.
        //aa-> numero de dos dígitos del ValorDireccion
        bufferSalidaIris[2]=0x30+(unsigned char)abs(DEC_decimal);
        bufferSalidaIris[3]=0x30+(unsigned char)abs(DEC_unidad);

        //envió de valor de Focus
        serialPort_BUS_CAM->Write(bufferSalidaIris,0,5);

}//Evento_IrisOperation()
```

4.2.7. Comando para la selección de la memoria de Pre-set

Comando para seleccionar el número de memoria de Pre-set. Podemos seleccionar con este protocolo hasta 50 Pre-set. El rango del número de Pre-set es de 1 a 49.

Sintaxis → (**#Maab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
M : Identificador de datos ID de información -> establece el operador 'M' del comando SET_MEMORY..
a a : Valor numérico de dos dígitos que indica el numero de Pre-set seleccionado(00..49).
b : Delimitador (Current-CR)

Ejemplo: Para '#M00.', en Hexa. es → '23 4D 30 30 0D'

4.2.8. Comando para guardar una posición en la memoria de Pre-set

Comando para guardar los ajustes de posición y lentes en un número de memoria de Pre-set determinado. El rango del número de Pre-set es de 1 a 49.

Sintaxis → (**#Raab**)

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
R : Identificador de datos ID de información -> establece el operador 'R' del comando SAVE_IN_MEMORY..
a a : Valor numérico de dos dígitos que indica el numero de Pre-set seleccionado(00..49).
b : Delimitador (Current-CR)

Ejemplo: Para '#R00.', en Hexa. es → '23 52 30 30 0D'

<Código C++> ('CamCommands.h')

```
public: cli::array<unsigned char,1>^EventoCambioMem(cli::array<unsigned
char,1>^bufferEntradaPreset,System::IO::Ports::SerialPort^
serialPort_BUS_CAM,int numeroPreset_Decimal,int numeroPreset_Unidad) {
///   Función para seleccionar o guardar Pre-set Memory.
///   Variables:
///              array<unsigned char,1>^bufferEntradaPreset --> sirve para
indicar el valor de información id del bufferSalidaPreset[1],
///         que indica el modo de operación del evento: R-> cambio de Pre-set
///                                                                    M->
memorizar Pre-set
///              array<unsigned char,1>^bufferSalidaPreset --> devolvemos el
valor actual del bufferSailidaPreset
```

```
///               int numeroPreset_Decimal --> numero entero que indica el
decimal del numero de Pre-set, que deseamos cambiar o memorizar.
///               int numeroPreset_Unidad -->  numero entero que indica la
unidad del numero de Pre-set, que deseamos cambiar o memorizar.
///

      //buffer de salida de puertos
      array<unsigned char>^ bufferSalidaPreset= gcnew array<unsigned char>(5);

      //introducimos el número de preste seleccionado y el modo de operación del
      comando Pre-set.
      bufferSalidaPreset[0]=0x23; //Valor que indica comamndos del Pc
      bufferSalidaPreset[1]=bufferEntradaPreset[1]; //Valor de informacion ID
      bufferSalidaPreset[2]=0x30+(unsigned char)abs(numeroPreset_Decimal);
      bufferSalidaPreset[3]=0x30+(unsigned char)abs(numeroPreset_Unidad);
      bufferSalidaPreset[4]=0x0D; //Valor Delimiter (Current CR)

      //envió de valor del comando del Pre-set
      serialPort_BUS_CAM->Write(bufferSalidaPreset,0,5);

      //valor bit desactivación Grabar Pre-set -> MEMORY
      bufferSalidaPreset[1]='R';
      return bufferSalidaPreset;
}//EventoCambioMem()
```

4.2.9. Resultados de los comandos de memorias de Pre-set

Respuesta de la cabeza Pan/Tilt, cuando seleccionamos un Pre-set nos devuelve el resultado del Pre-set seleccionado.

Sintaxis → **(Saab)**

S : Identificador de datos ID de informacion -> Indica la respuesta del comando Pre-set.
a a : Valor numérico de dos dígitos que indica el numero de Pre-set establecido(00..49).
b : Delimitador (Current-CR)

4.2.10. Comando para el encendido (POWER ON/OFF)

Comando para el apagado y encendido de las Cámaras y el cabezal P/T.

Sintaxis → **(#Oab)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
O : Identificador de datos ID de informacion -> establece el operador 'O' del comando Power.
a : Valor numérico de un solo digito que indica si apagamos o encendemos
Valor PowerON -> '1'
Valor PowerOFF -> '0'
b : Delimitador (Current-CR)

Ejemplo: Para '#O1.', en Hexa. es → '23 4F 31 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void EventoPower(System::IO::Ports::SerialPort^
serialPort_BUS_CAM,unsigned char ValorPower){
/* Función para encender/apagar las Cámaras
ValorPower: PowerON -> '1'/ PowerOFF -> '0'
*/
      //Variable POWER-> #Oa.
      array<unsigned char>^bufferSalidaPower=gcnew array<unsigned char>(4);
```

```
        //Valor inicial Power ON/OFF
        bufferSalidaPower[0]=0x23;
        bufferSalidaPower[1]=0x4F;
        bufferSalidaPower[2]=0x31; // Valor PowerON -> '1'/ PowerOFF -> '0'
        bufferSalidaPower[3]=0x0D;//delimiter (Current-CR)

        //Introduce el valorPower: PowerON -> '1'/ PowerOFF -> '0'
        bufferSalidaPower[2]=ValorPower;

        //Enviamos el comando Power
        serialPort_BUS_CAM->Write(bufferSalidaPower,0,4);

}//Power()
```

4.2.11. Comando para establecer los controles de Condición

Comando para la condición de los nueve interruptores de los controles de condición. Establece para un interruptor dado su estado -> 0->OFF, 1->ON

Sintaxis → **(#Dabc)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
D : Identificador de datos ID de información -> establece el operador 'D' del comando Condition Control.
a b : Numero de interruptor y valor de estado Activado/desactivado.

Interruptor	Numero (Byte a)	Estado (Byte b)
EXT	1	0:Off / 1: On
ND	2	0:Off / 1: On
IRIS_MODE	3	0:Off / 1: On
LAMP	4	0:Off / 1: On
INQUIRY_of_FILAMENT_BREAK: (SW_Status)->NONE	5	NONE
SERVICE_SWITCH	6	0:Off / 1: On
DEF	7	0:Off / 1: On
WIPE	8	0:Off / 1: On
HEAT/FAN	9	0:Off / 1: On

c : Delimitador (Current-CR)

Ejemplo: Para '#D11.', en Hexa. es → '23 44 31 31 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void
Evento_ConditionControl(System::IO::Ports::SerialPort^serialPort_BUS_CAM,cli::array<unsigned char,1>^bufferSalidaConditionControl,int SW_Number,int SW_Status){
/*Comando para la condición de control --> establece para un interruptor su estado ON/OFF
     SW_Number --> selecciona un interruptor del 1..9:
     1->EXT
     2->ND
     3->IRIS_MODE
     4->LAMP
     5->INQUIRY_of_FILAMENT_BREAK: (SW_Status)->NONE
     6->SERVICE_SWITCH
     7->DEF
     8->WIPE
     9->HEAT/FAN
```

```
SW_Status --> Establece el estado del interruptor seleccionado: 0->OFF, 1->ON*/

      //Variable ConditionControl-> #Dab.
      bufferSalidaConditionControl=gcnew array<unsigned char>(5);

      //Valor inicial ConditionControl
      bufferSalidaConditionControl[0]=0x23;
      bufferSalidaConditionControl[1]=0x44;
      bufferSalidaConditionControl[2]=0x30;
      bufferSalidaConditionControl[3]=0x30;
      bufferSalidaConditionControl[4]=0x0D;//delimiter (Current-CR)

      //guardamos la informacion de ConditionControl
      bufferSalidaConditionControl[2]=0x30+(unsigned char)abs(SW_Number);
      bufferSalidaConditionControl[3]=0x30+(unsigned char)abs(SW_Status);

      //Enviamos el comando ConditionControl
      serialPort_BUS_CAM->Write(bufferSalidaConditionControl,0,5);

}//Evento_ConditionControl()
```

4.2.12. Resultado del comando del control de condición

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando condición control nos devuelve el resultado del interruptor seleccionado y su estado: Activado/Desactivado.

Sintaxis → **(dabc)**

d: Identificador de datos ID de informacion -> Identificador de la respuesta del comando Condition Control 'd'.

a : Valor numérico de 1 a 9 que indica el numero de interruptor establecido.

b: Valor numérico que muestra el estado del interruptor seleccionado (0->OFF, 1->ON).

c : Delimitador (Current-CR)

4.2.13. Comando para Establecer/ Resetear el límite de posición

Comando para el establecer o resetear los límites de rotación de la cabeza P/T.

Sintaxis → **(#Lab)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.

L : Identificador de datos ID de informacion -> establece el operador 'L' del comando LIMIT.

a : Valor numérico del 1 al 4 que indica cual es el límite establecido:

ModeLimit -> 1:Tilt Upper Limit
2:Tilt Under Limit
3:Pan Left Limit
4:Pan Right Limit

b : Delimitador (Current-CR)

Ejemplo: Para '#L1.', en Hexa. es → '23 4C 31 0D'

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_SettingPositionLimit(System::IO::Ports::SerialPort^
serialPort_BUS_CAM,int ModeLimit){
/*Comando para Establecer o resetear el límite de posición.
ModeLimit -> 1:Tilt Upper Limit
                  2:Tilt Under Limit
                  3:Pan Left Limit
                  4:Pan Right Limit */
```

```
        //Variable SettingPositionLimit-> #La.
        array<unsigned char>^bufferSalidaSettingPositionLimit=gcnew array<unsigned
char>(4);

        //Valor inicial SettingPositionLimit
         bufferSalidaSettingPositionLimit[0]=0x23;
         bufferSalidaSettingPositionLimit[1]=0x4C;
         bufferSalidaSettingPositionLimit[2]=0x30;
         bufferSalidaSettingPositionLimit[3]=0x0D;//delimiter (Current-CR)

        //Introducimos el valor ModeLimit
         bufferSalidaSettingPositionLimit[2]=0x30+(unsigned char)abs(ModeLimit);

        //Enviamos el comando SettingPositionLimit
        serialPort_BUS_CAM->Write(bufferSalidaSettingPositionLimit,0,4);

}//Evento_SettingPositionLimit()
```

4.2.14. Resultados del comando para Establecer/ Resetear el límite de posición

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando Limit nos devuelve el resultado del estado del límite: Activado/Desactivado.

Sintaxis → **(lab)**

l: Identificador de datos ID de informacion -> Identificador de la respuesta del comando Limit 'l'.

a : Valor numérico que muestra la condición del límite (0->Reset, 1->Setting).

b : Delimitador (Current-CR)

4.2.15. Comandos para establecer las posiciones Pan y Tilt

Comando para establecer una determinada posición pan/tilt en grados.

En la siguiente tabla se muestra el rango en grados de los valores de posición Pan/Tilt, con sus valores correspondientes en decimal y en hexadecimal.

	RANGO	Valor [Hex.]	Valor [Decimal]
PAN	0º..300º	0000..FFFF	0..65535
TILT	0º..190º	0000..FFFF	0..65535

Sintaxis→**(#Uaaaabbbbcd)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.

U : Identificador de datos ID de informacion -> establece el operador 'U' del comando Setting P/T position.

a a a a : Valor numérico de 4 dígitos en Hexadecimal que indica la posición Pan.

b b b b : Valor numérico de 4 dígitos en Hexadecimal que indica la posición Tilt

c : Check Sum -> suma de verificación (es la suma de longitud=1 byte, de los 8 bytes de datos de posición)

ChkS= HEX(a+a+a+a+b+b+b+b) [Byte]

d : Delimitador (Current-CR)

Ejemplo: Para '#U00000000Ç.', en Hexa. es → '23 55 30 30 30 30 30 30 30 30 80 0D'

NOTA: Esta función no está habilitada para el modelo de cabeza P/T: AW-PH300A

<Código C++> (‘CamCommands.h’)

```
public: array<unsigned char,1>^  Evento_SetPosition
(System::IO::Ports::SerialPort^ serialPort_BUS_CAM, int PanPosition_grados, int
TiltPosition_grados){
/*Comando para establecer una determinada posición pan/tilt en grados y devuelve
el valor de la posición establecida
array<unsigned char,1>^bufferSalidaSettingPosition

variables:
            int PanPosition_grados-> Posición panorámica en grados 0°..300°
            int TiltPosition_grados-> Posición de inclinación en grados 0°..190°
            array<unsigned char,1>^bufferSalidaSettingPosition -> valor del
      comando completo que establece la posición pan/tilt.
*/

      //Declaramos la variable bufferSalidaSettingPosition -> #U000000000.
      array<unsigned char>^bufferSalidaSettingPosition=gcnew array<unsigned
char>(12);

      //Valor inicial bufferSalidaSettingPosition -> #U000000000.
      bufferSalidaSettingPosition[0]=0x23;
      bufferSalidaSettingPosition[1]=0x55;
      bufferSalidaSettingPosition[2]=0x30;//Bytes aaaa -> PanPosition
      bufferSalidaSettingPosition[3]=0x30;
      bufferSalidaSettingPosition[4]=0x30;
      bufferSalidaSettingPosition[5]=0x30;
      bufferSalidaSettingPosition[6]=0x30;//Bytes bbbb -> TiltPosition
      bufferSalidaSettingPosition[7]=0x30;
      bufferSalidaSettingPosition[8]=0x30;
      bufferSalidaSettingPosition[9]=0x30;
      bufferSalidaSettingPosition[11]=0x0D;//delimiter (Current-CR)
      for(int i=2;i<10;i++){ //Byte d -> checkSum 8 byte data
            bufferSalidaSettingPosition[10]=bufferSalidaSettingPosition[10]+buff
      erSalidaSettingPosition[i];
      }//for()

      int PanPosition;
      int TiltPosition;

      int PanPosition_Hex;
      int TiltPosition_Hex;

      //Pasamos de grados a decimal
       PanPosition=PanPosition_grados*218;
       TiltPosition=TiltPosition_grados*344;

//Pasamos de decimal a hexadecimal y guardamos en bufferSalidaSettingPosition
        for(int i=5;i>=2;i--){
              PanPosition_Hex=PanPosition%16;
              if(PanPosition_Hex<=9){
                    bufferSalidaSettingPosition[i]=0x30+(unsigned
char)(PanPosition_Hex);
             }else{
                    bufferSalidaSettingPosition[i]=0x37+(unsigned
char)(PanPosition_Hex);
             }//if()
             PanPosition=PanPosition/16;

      }//for()

      for(int i=9;i>=6;i--){
        TiltPosition_Hex=TiltPosition%16;
        if(TiltPosition_Hex<=9){
             bufferSalidaSettingPosition[i]=0x30+(unsigned
char)(TiltPosition_Hex);
```

```
        }else{
              bufferSalidaSettingPosition[i]=0x37+(unsigned
char)(TiltPosition_Hex);
        }//if()
        TiltPosition=TiltPosition/16;

      }//for()

      //Calculo del valor CheckSum
      bufferSalidaSettingPosition[10]=NULL;

      for(int i=2;i<10;i++){

            bufferSalidaSettingPosition[10]=bufferSalidaSettingPosition[10]+buff
      erSalidaSettingPosition[i];
      }//for()

      //Enviamos el comando SettingPosition
      serialPort_BUS_CAM->Write(bufferSalidaSettingPosition,0,12);

      //Devolvemos el valor de bufferSalidaSettingPosition
      return bufferSalidaSettingPosition;
}//Evento_SetPosition()
```

4.2.16. Resultado para el comando que establece una posición Pan/Tilt

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando para establecer las posiciones Pan y Tilt nos devuelve el resultado de la posición establecida P/T.

Sintaxis → (uaaaabbbbcd)

u: Identificador de datos ID de informacion -> Identificador de la respuesta del comando para establecer las posiciones Pan y Tilt 'u'.

a a a a : Valor numérico de 4 dígitos en Hexadecimal que indica la posición Pan.

b b b b : Valor numérico de 4 dígitos en Hexadecimal que indica la posición Tilt

c : Check Sum -> suma de verificación (es la suma de longitud=1 byte, de los 8 bytes de datos de posición)

ChkS= HEX(a+a+a+a+b+b+b+b) [Byte]

d : Delimitador (Current-CR)

NOTA: Esta función no está habilitada para el modelo de cabeza P/T: AW-PH300A

4.2.17. Comando para establecer la condición de las lentes

Comando para establecer una determinada posición de las lentes (Zoom, Focus e Iris).

En la siguiente tabla se muestra el rango en porcentajes de los valores de posición de las lentes con sus valores correspondientes en decimal y en hexadecimal.

	RANGO	Valor [Hex.]	Valor [Decimal]
ZOOM	0..99	000..FFF	0..4095
FOCUS	0..99	000..FFF	0..4095
IRIS	0..99	000..FFF	0..4095

Sintaxis→**(#Aaaabbbcccde)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.

A : Identificador de datos ID de informacion -> establece el operador 'A' del comando Setting Lens Condition.

a a a : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Zoom.

b b b : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Focus

c c c : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Iris

d: Check Sum -> suma de verificación (es la suma de longitud=1 byte, de los 9 bytes de datos de la posición de las lentes)

ChkS= HEX(a+a+a+b+b+b+c+c+c) [Byte]

e : Delimitador (Current-CR)

Ejemplo: Para '#A00000000Ç.', en Hexa. es → '23 41 30 30 30 30 30 30 30 80 0D'

NOTA: Esta función no está habilitada para el modelo de cabeza P/T: AW-PH300A

<Código C++> ('CamCommands.h')

```
public: cli::array<unsigned char,1>^ Evento_LensCondition
(System::IO::Ports::SerialPort^  serialPort_BUS_CAM,unsigned int ZoomPosition,
unsigned int FocusPosition, unsigned int IrisPosition){
/* Comando para establecer una determinada posición de las lentes: Zoom, Focus y
Iris y devuelve el valor del comando que establece la condición de las lentes
variables:
      unsigned int ZoomPosition-> Valor de la posición del Zoom 0..99
      unsigned int FocusPosition-> Valor de la posición del Foco 0..99
      unsigned int IrisPosition-> Valor de la posición del Iris 0..99
      array<unsigned char>^bufferSalidaLensCondition-> Valor de salida del
comando para establecer las condiciones de las lentes */

      //Declaramos la variable bufferSalidaLensCondition -> #A000000000d.
      array<unsigned char>^bufferSalidaLensCondition=gcnew array<unsigned
char>(13);

      //Valor inicial bufferSalidaLensCondition -> #A000000000d.
      bufferSalidaLensCondition[0]=0x23;
      bufferSalidaLensCondition[1]=0x41;
      bufferSalidaLensCondition[2]=0x30;//Bytes aaa -> ZoonPosition
      bufferSalidaLensCondition[3]=0x30;
      bufferSalidaLensCondition[4]=0x30;
      bufferSalidaLensCondition[5]=0x30;//Bytes bbb -> FocusPosition
      bufferSalidaLensCondition[6]=0x30;
      bufferSalidaLensCondition[7]=0x30;
      bufferSalidaLensCondition[8]=0x30;//Bytes ccc -> IrisPosition
      bufferSalidaLensCondition[9]=0x30;
      bufferSalidaLensCondition[10]=0x30;
      bufferSalidaLensCondition[12]=0x0D;//delimiter (Current-CR)
      for(int i=2;i<=10;i++){ //Byte d -> checkSum 8 byte data

      bufferSalidaLensCondition[11]=bufferSalidaLensCondition[11]+bufferSalidaLe
nsCondition[i];
      }//for()

      unsigned int ZoomPosition_Hex;
      unsigned int FocusPosition_Hex;
      unsigned int IrisPosition_Hex;

      //DesNormalizamos los valores Zoom,focus y iris.
       ZoomPosition=ZoomPosition*41;
       FocusPosition=FocusPosition*41;
       IrisPosition=IrisPosition*41;
```

```
      //Pasamos de decimal a hexadecimal y guardamos en
bufferSalidaSettingPosition

        for(int i=4;i>=2;i--){
              ZoomPosition_Hex=ZoomPosition%16;
              if(ZoomPosition_Hex<=9){
                    bufferSalidaLensCondition[i]=0x30+(unsigned
char)(ZoomPosition_Hex);
              }else{
                    bufferSalidaLensCondition[i]=0x37+(unsigned
char)(ZoomPosition_Hex);
              }//if()
              ZoomPosition=ZoomPosition/16;

      }//for()

      for(int i=7;i>=5;i--){
              FocusPosition_Hex=FocusPosition%16;
              if(FocusPosition_Hex<=9){
                    bufferSalidaLensCondition[i]=0x30+(unsigned
char)(FocusPosition_Hex);
              }else{
                    bufferSalidaLensCondition[i]=0x37+(unsigned
char)(FocusPosition_Hex);
              }//if()
              FocusPosition=FocusPosition/16;

      }//for()

      for(int i=10;i>=8;i--){
              IrisPosition_Hex=IrisPosition%16;
              if(IrisPosition_Hex<=9){
                    bufferSalidaLensCondition[i]=0x30+(unsigned
char)(IrisPosition_Hex);
              }else{
                    bufferSalidaLensCondition[i]=0x37+(unsigned
char)(IrisPosition_Hex);
              }//if()
              IrisPosition=IrisPosition/16;

      }//for()

      //Calculo del valor CheckSum
            bufferSalidaLensCondition[11]=NULL;

      for(int i=2;i<=10;i++){

            bufferSalidaLensCondition[11]=bufferSalidaLensCondition[11]+bufferSa
      lidaLensCondition[i];
      }//for()

      //Enviar comando LensCondition
      serialPort_BUS_CAM->Write(bufferSalidaLensCondition,0,13);

      //Devolvemos el valor de bufferSalidaLensCondition
      return bufferSalidaLensCondition;
}//Evento_LensCondition()
```

4.2.18. Resultado del comando establecer condiciones de las lentes

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando para establecer las condiciones de las lentes nos devuelve el resultado de la posición de las mismas.

Sintaxis → **(abbbcccdddef)**

a: Identificador de datos ID de informacion -> Identificador de la respuesta del comando para establecer las posiciones Pan y Tilt 'a'.
b b b : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Zoom.
c c c : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Focus.
d d d : Valor numérico de 3 dígitos en Hexadecimal que indica la posición Iris.
e: Check Sum -> suma de verificación (es la suma de los 9 bytes de datos de condición de las lentes)
ChkS= HEX(b+b+b+c+c+c+d+d+d) [Byte]
f : Delimitador (Current-CR)

NOTA: Esta función no está habilitada para el modelo de cabeza P/T: AW-PH300A

4.2.19. Comando para la monitorización de las condiciones Pan/Tilt

Comando para establecer la configuración de la respuesta del Pan/Tilt Condition del cabezal Pan/tilt.

Sintaxis→ **(#Bcbade)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.
B: Identificador de datos ID de informacion -> establece el operador 'B' del comando para Monitorización las condiciones del P/T.
a : Valor numérico de un byte de informacion (en Hexadecimal)
bit[0] : respuesta para el valor AD del pan (ON : 1, OFF : 0)
bit[1] : respuesta para el valor de los comandos pan (ON : 1, OFF : 0)
bit[3] :respuesta para el valor de desviación pan (ON : 1, OFF : 0)
bit[7] : respuesta para el valor íntegro pan (ON : 1, OFF : 0)
b : Valor numérico de un byte de informacion (en Hexadecimal)
bit[0] : respuesta para el valor diferencial pan (ON : 1, OFF : 0)
bit[1] : respuesta para el valor DA pan (ON : 1, OFF : 0)
bit[3] : respuesta para el valor AD tilt (ON : 1, OFF : 0)
bit[7] : respuesta para el valor comandos tilt (ON : 1, OFF : 0)
c : Valor numérico de un byte de informacion (en Hexadecimal)
bit[0] : respuesta para el valor de desviación Tilt (ON : 1, OFF : 0)
bit[1] : respuesta para el valor íntegro tilt (ON : 1, OFF : 0)
bit[3] : respuesta para el valor diferencial tilt (ON : 1, OFF : 0)
bit[7] : respuesta para el valor DA tilt (ON : 1, OFF : 0)
d : c : Check Sum -> suma de verificación (es la suma de longitud=1 byte, de los 3 bytes de datos de informacion)
ChkS= HEX(a+b+c) [Byte]
e : Delimitador (Current-CR)

Ejemplo: Para '#B111.', en Hexa. es → '23 42 31 31 31 93 0D'

NOTA: Esta función no está habilitada para el modelos de cabeza P/T: AW-PH300A y AW-PH360L

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_MonitoringPanTiltCondition
(System::IO::Ports::SerialPort^  serialPort_BUS_CAM){
/*Comando para establecer la configuración de la respuesta del Pan/Tilt
Condition del Pan/tilt Head.

*/
      //Variable MonitoringPanTiltCondition -> #B c b a d .
       array<unsigned char>^bufferSalidaMonitoringPanTiltCondition=gcnew
array<unsigned char>(7);

      //Valor inicial MonitoringPanTiltCondition -> #Bcbad.
      bufferSalidaMonitoringPanTiltCondition[0]=0x23;
      bufferSalidaMonitoringPanTiltCondition[1]=0x42;
      bufferSalidaMonitoringPanTiltCondition[2]=0x00;
      bufferSalidaMonitoringPanTiltCondition[3]=0x80;
      bufferSalidaMonitoringPanTiltCondition[4]=0x02;
      bufferSalidaMonitoringPanTiltCondition[6]=0x0D;
      for(int i=2;i<=4;i++){ //Byte d -> checkSum 8 byte data

            bufferSalidaMonitoringPanTiltCondition[5]=bufferSalidaMonitoringPanT
      iltCondition[5]+bufferSalidaMonitoringPanTiltCondition[i];
      }//for()

      //envió de valor de bufferSalidaMonitoringPanTiltCondition
       serialPort_BUS_CAM->Write(bufferSalidaMonitoringPanTiltCondition,0,7);
}
```

4.2.20. Comando para la respuesta de monitorización de las condiciones Pan/Tilt

Comando para la petición de la repuesta de la monitorización de la condiciones Pan/Tilt. Cuando ejecutamos este comando recibimos el resultado de las condiciones Pan/Tilt.

Sintaxis → **(#Ja)**

 # : Identificador de datos ID -> Indica que los datos proceden desde el PC.

J : Identificador de datos ID de informacion -> establece el operador 'J' del comando para la petición de resultados de las condiciones Pan/Tilt

a : Delimitador (Current-CR)

Ejemplo: Para '#J.', en Hexa. es → '23 4A 0D'

NOTA: Esta función no está habilitada para el modelos de cabeza P/T: AW-PH300A y AW-PH360L

<Código C++> ('CamCommands.h')

```
public: System::Void Evento_MonitoringPanTiltConditionRequest
(System::IO::Ports::SerialPort^  serialPort_BUS_CAM){
/*Comando para la petición de la respuesta por parte del Pan/Tilt Head,
para recibir el comando para la monitorización de las condiciones Pan/Tilt
*/
      //Variable MonitoringPanTiltConditionRequest -> #J.
       array<unsigned char>^bufferSalidaMonitoringPanTiltConditionRequest=gcnew
array<unsigned char>(3);

      //Valor inicial MonitoringPanTiltConditionRequest -> #J.
       bufferSalidaMonitoringPanTiltConditionRequest[0]=0x23;
       bufferSalidaMonitoringPanTiltConditionRequest[1]=0x4A;
       bufferSalidaMonitoringPanTiltConditionRequest[2]=0x0D;
```

```
      //envió de valor de bufferSalidaMonitoringPanTiltConditionRequest
serialPort_BUS_CAM->Write(bufferSalidaMonitoringPanTiltConditionRequest,0,3);

}
```

4.2.21. Resultado del comando para la petición de respuesta de monitorización de las condiciones de la cabeza P/T

Respuesta de la cabeza Pan/Tilt, cuando ejecutamos el comando para la petición de respuesta de monitorización de las condiciones Pan/Tilt, nos devuelve el resultado de las condiciones P/T que hemos habilitado.

Sintaxis → **(Jaaaabbbbccccddddeeeeffffgggghhhhiiiikkkkllllmmmmn)**

J : Identificador de datos ID de informacion -> Identificador de la respuesta del comando para monitorización de los resultados Pan y Tilt 'J'.

a a a a : numérico de 4 dígitos en Hexadecimal que indica el valor pan AD (0000-FFFF).

b b b b : numérico de 4 dígitos en Hexadecimal que indica el valor para los comandos pan (0000-FFFF).

c c c c : numérico de 4 dígitos en Hexadecimal que indica el valor de desviación pan (0000-FFFF).

d d d d : numérico de 4 dígitos en Hexadecimal que indica el valor integro pan (0000-FFFF).

e e e e : numérico de 4 dígitos en Hexadecimal que indica el valor pan diferencial (0000-FFFF).

f f f f numérico de 4 dígitos en Hexadecimal que indica el valor pan DA (0000-FFFF).

g g g g : numérico de 4 dígitos en Hexadecimal que indica el valor tilt AD (0000-FFFF).

h h h h : numérico de 4 dígitos en Hexadecimal que indica el valor para los comandos tilt (0000-FFFF).

i i i i : numérico de 4 dígitos en Hexadecimal que indica el valor de desviación tilt (0000-FFFF).

k k k k : numérico de 4 dígitos en Hexadecimal que indica el valor integro tilt (0000-FFFF).

l l l l : numérico de 4 dígitos en Hexadecimal que indica el valor diferencial tilt (0000-FFFF).

m m m m : numérico de 4 dígitos en Hexadecimal que indica el valor tilt DA (0000-FFFF).

n : Delimitador (Current-CR)

NOTA: Esta función no está habilitada para el modelos de cabeza P/T: AW-PH300A y AW-PH360L

4.3. Comparación de protocolos

En este apartado vamos a indicar las principales diferencias que hay entre protocolos de comunicación, en la siguiente tabla se muestra un resumen de ello:

	Protocolo RS-422	Protocolo RS-232C
Tipo de Comunicación	**Secuencia periódica sincronizada de comandos**	**Por Eventos**
Buses de comunicación	**Tres buses:** **BUS_TX_P/T** **BUS_RX_P/T** **BUS_TALLY**	**Dos Buses:** **BUS_TX** **BUS_RX**
Número de Pre-sets	**10 Pre-sets**	**50 Pre-sets**
Comando White Balance	**SI**	**NO**
Comando SPEED	**SI**	**NO**
Comandos para establecer la Posición P/T y LENTES	**NO**	**SI** **(Solo para el modelo AW-360L)**

El resto de funciones de los protocolos están incluidas por igual en ambos, pero hay diferencias en su sintaxis.

Además en el protocolo RS-422 hay unos comandos extra para controlar la comunicación secuencial de comandos, como comandos para: el corte de línea (Cut Off line) y la detección de errores.

También hay diferencia relacionada con los comandos de operación P/T, lentes e interruptores de condición. En el protocolo RS-422 estos comandos están contenidos en un solo comando 'Command Status', en cambio en el protocolo RS-232C están incluidos en comandos diferentes.

5. Interface

5.1. Interface RS-422: Del Hub al PC

Al tener una comunicación periódica y continua entre el panel y el Hub, debemos plantearnos el diseño de un conjunto de dispositivos que nos permitan cambiar el control del Panel al PC, sin que se produzca ningún corte en la transmisión y recepción de datos entre estos dispositivos y el Hub.

Pero el inconveniente de esto es tener que utilizar una tarjeta remota de relés para conmutar entre el panel y el PC. Esto provocaba la desconexión temporal del panel y por lo tanto la inhabilitación del mismo.

La ventaja de este tipo de interface es que nos evitamos tener que tirar un cable hacia cada cámara. Ya que nos conectamos por medio la conexión "P/T Control IN" entre el panel y el Hub.

Conversores:

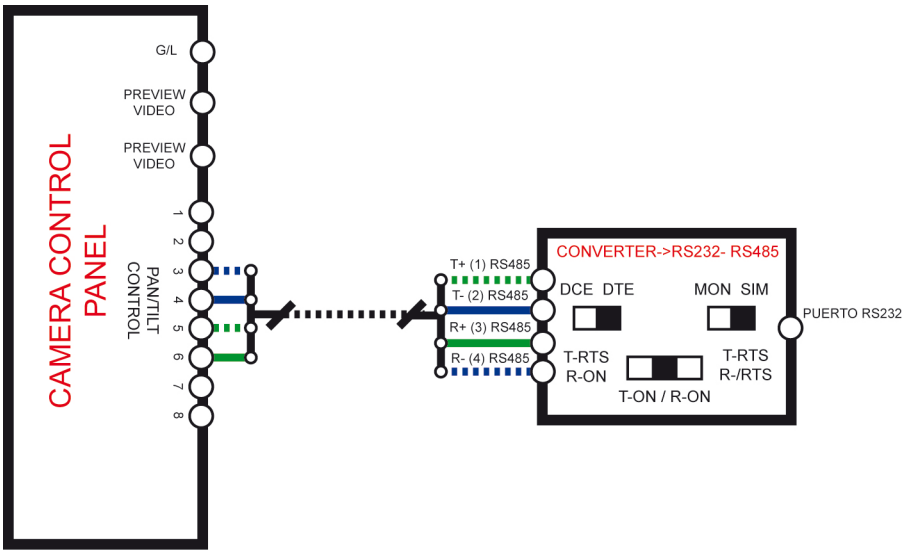
Lo primero, tenemos que convertir la señales RS-422 de los Buses "Pan/Tilt" y "Tally" al protocolo RS-232, tanto del Hub como del Panel. Para ello, vamos a utilizar conversores "RS-485 a RS-232" (ic485-lp Multi-Drop Interface Converter).

Fig. 1: "Imagen del conversor i485-lp"

Para el BUS_P/T del Panel y del Hub necesitaremos 2 conversores i485-lp, porque este Bus es bidireccional (es decir, de transmisión y recepción). En cambio para el Bus de Tally solamente necesitaremos un conversor i485-lp, ya que el bus es unidireccional (solamente transmite desde el panel).

La configuración y las conexiones de los conversores son las siguientes tal y como se muestran a continuación:

- **Para el panel:** el conversor debe estar configurado en modo "Equipo terminal de datos", es decir como fuente del sistema o emisor:

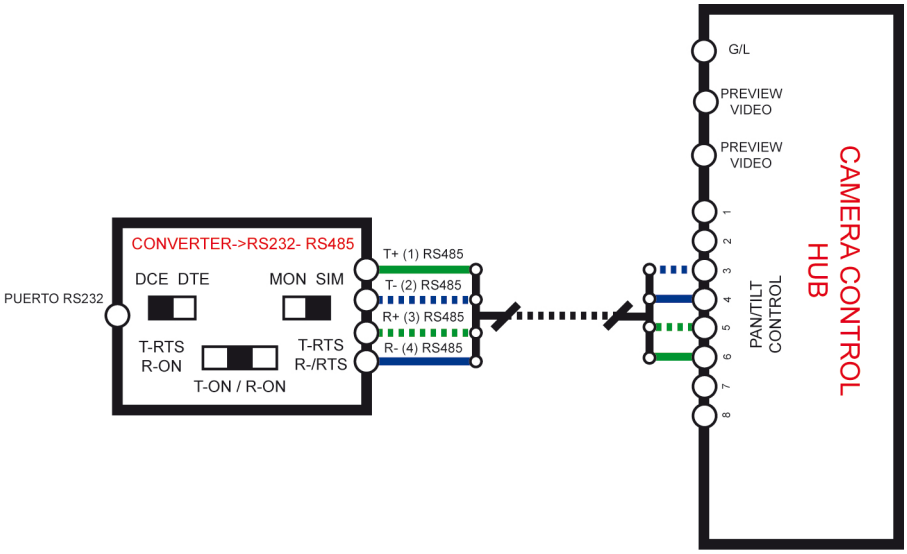


Configuración Conversor
DTE
SIM
T-ON/R-ON

Fig. 2: "Configuración y conexiones del conversor i485lp para el panel"

Tabla de conexión (Panel – Conversor)			
Panel	**SEÑAL DE CONTACTO**	**COLOR DEL CABLE**	**Conversor i485-lp**
Pin 3	COLD 2	Azul-Blanco	Rx-(pin 4) RS-485
Pin 4	COLD 1	Azul	Tx-(pin 2) RS-485
Pin 5	HOT 1	Verde-Blanco	Tx+(pin 1) RS-485
Pin 6	HOT 2	Verde	Rx+(pin 3) RS-485

- **Para el Hub:** el conversor debe de estar configurado en modo "Equipo de Comunicación de datos"(receptor):

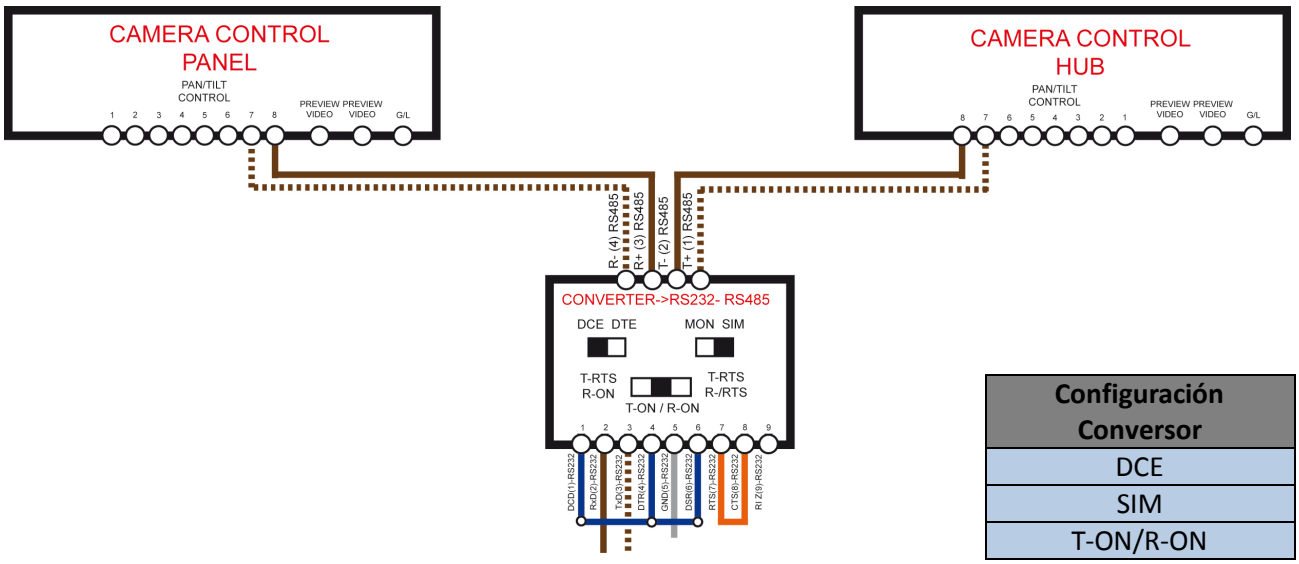


Configuración Conversor
DCE
SIM
T-ON/R-ON

Fig. 3: "Configuración y conexiones del conversor i485lp para el Hub"

Tabla de conexión (Panel – Conversor)			
Panel	**SEÑAL DE CONTACTO**	**COLOR DEL CABLE**	**Conversor i485-lp**
Pin 3	COLD 2	Azul-Blanco	Tx-(pin 2) RS-485
Pin 4	COLD 1	Azul	Rx-(pin 4) RS-485
Pin 5	HOT 1	Verde-Blanco	Rx+(pin 3) RS-485
Pin 6	HOT 2	Verde	Tx+(pin 1) RS-485

- Para el bus de Tally, configuraremos el conversor en modo “Equipo de Comunicación de datos” (receptor). Y conectaremos el panel y el Hub a través de él.



Configuración Conversor
DCE
SIM
T-ON/R-ON

Fig. 4: “Configuración y conexiones del conversor i485lp para el Bus de Tally entre el Hub y el Panel”

Tabla de conexión Tally (Panel)			
Panel	**SEÑAL DE CONTACTO**	**COLOR DEL CABLE**	**Conversor i485-lp**
Pin 7	Tally input 1	Marrón-Blanco	Rx-(pin 4) RS-485
Pin 8	Tally input 2	Marrón	Rx+(pin 3) RS-485

Tabla de conexión Tally (Hub)			
Panel	**SEÑAL DE CONTACTO**	**COLOR DEL CABLE**	**Conversor i485-lp**
Pin 7	Tally input 1	Marrón-Blanco	Tx+(pin 1) RS-485
Pin 8	Tally input 2	Marrón	Tx-(pin 2) RS-485

Tarjeta de control remoto de Relés (8-CHANNEL REMOTE RELAY CARD):

Para la conmutación entre el Hub y el Panel o PC, utilizaremos una tarjeta de relés. Esta tarjeta está controlada por medio del puerto RS-232 del PC. Posee 8 canales de reles por los cuales conmutaremos los buses de comunicación del panel y del PC con el Hub.

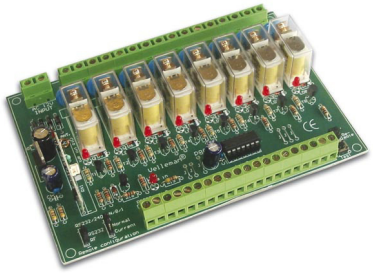


Fig. 4: "Imagen de 8-CHANNEL REMOTE RELAY CARD"

La Tarjeta de relés contiene una serie de instrucciones que utilizaremos para controlar el estado de los relés por medio del puerto del PC. Estas instrucciones están formadas por la siguiente secuencia de comandos:

1. Char($13) -> Hex(0x0D) -> ASCII(.)
2. Dirección de la placa (1..255)
3. Instrucción

Tabla de Instrucciones	
Instrucción	**Definición**
E	Parada de Emergencia
D	Reproducir la dirección
S	Activa un relé (1..8; el 9 activa todos los relés)
C	Desactiva un relé (1..8; el 9 activa todos los relés)
T	Función Toogle (seguido de un nº de relé 1..8)
A	Cambia la dirección actual de la placa (1..255)
F	Coloca todas las placas en la misma dirección (1)
B	Envía un Byte y establece en una sola instrucción el estado de todos los relés (MSB: relé 1; LSB: relé 8)

4. Dirección (1..255) o número de relé en ASCII(1..9)
5. CheckSum -> Suma de control (equivale a la suma en complemento 2 de los 4 bytes precedentes+1) .

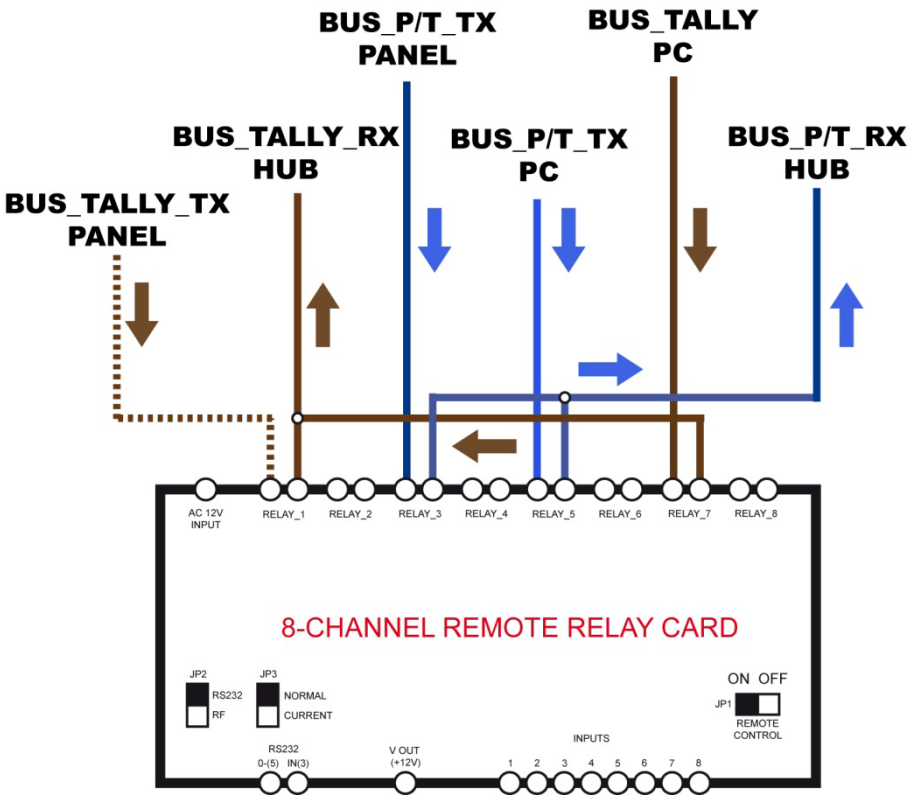
$$\mathrm{ChK} = \text{Ca2 (Byte 1+Byte2+Byte3+Byte4) +1}$$

Ejemplo de instrucción:

Para activar los reles:1 y 2 dejando los demás inactivos-> ASCII ("..B¬v")-> Hex(0D 01 42 C0 F0)

CheckSum= ca2 (0D+01+42+C0) + 1= ca2(10000)+1=0111111111+1= F0

Conectaremos cada una de las señales provenientes del panel, del Hub y del PC tal y como muestra el siguiente diagrama:



Otra cosa a tener en cuenta es el retardo que se produce al abrir o cerrar los relés que es de 375ms, desde que se envía el comando para cambiar el estado de los relés hasta que se produce el cambio.

Sabiendo todo esto ahora ya podemos comprender mejor el diagrama completo del interface. A continuación vamos a mostrar paso a paso las conexiones para cada bus y después el montaje completo.

Diagrama de Conexiones BUS_P/T:

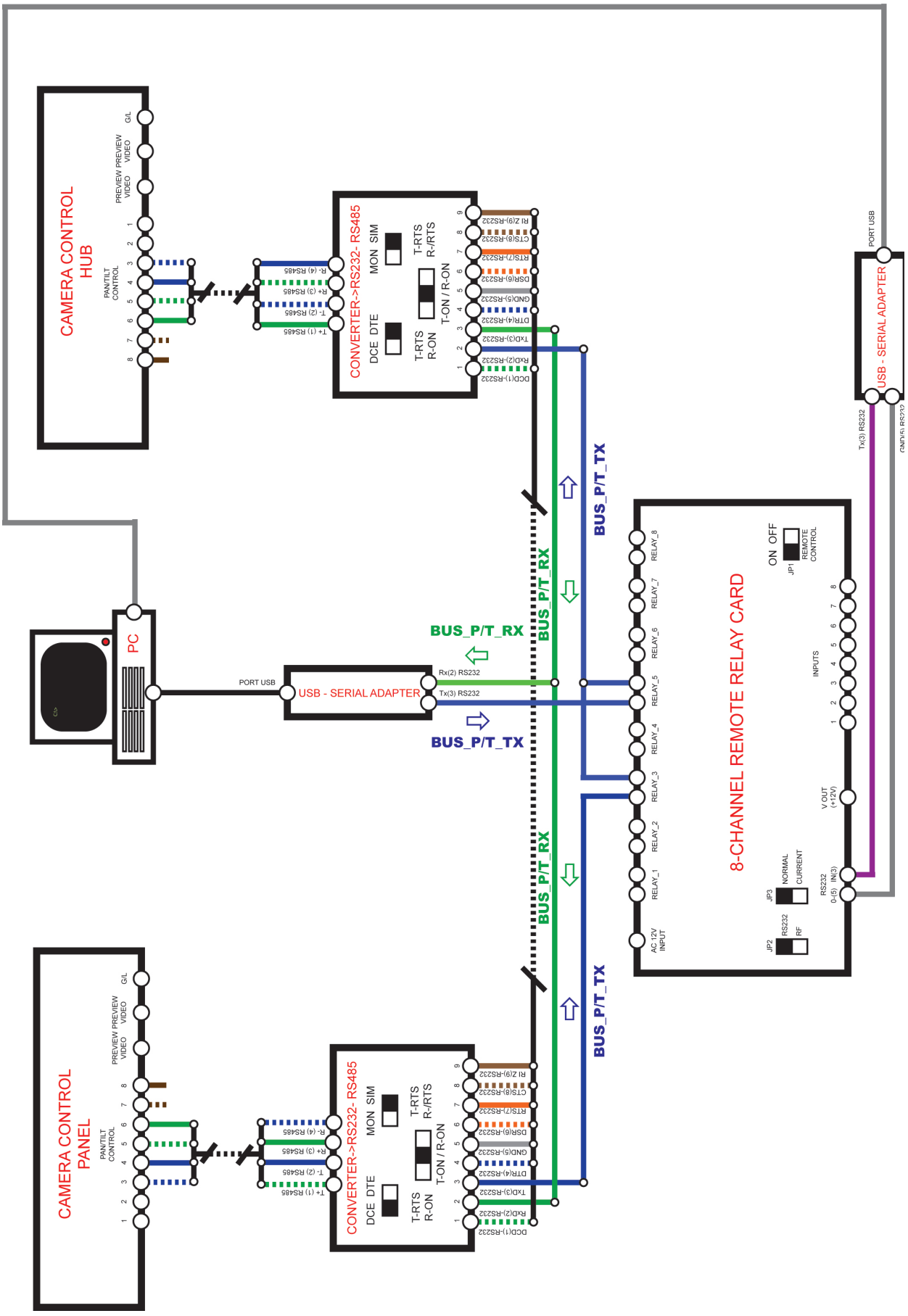


Diagrama de conexiones BUS_TALLY:

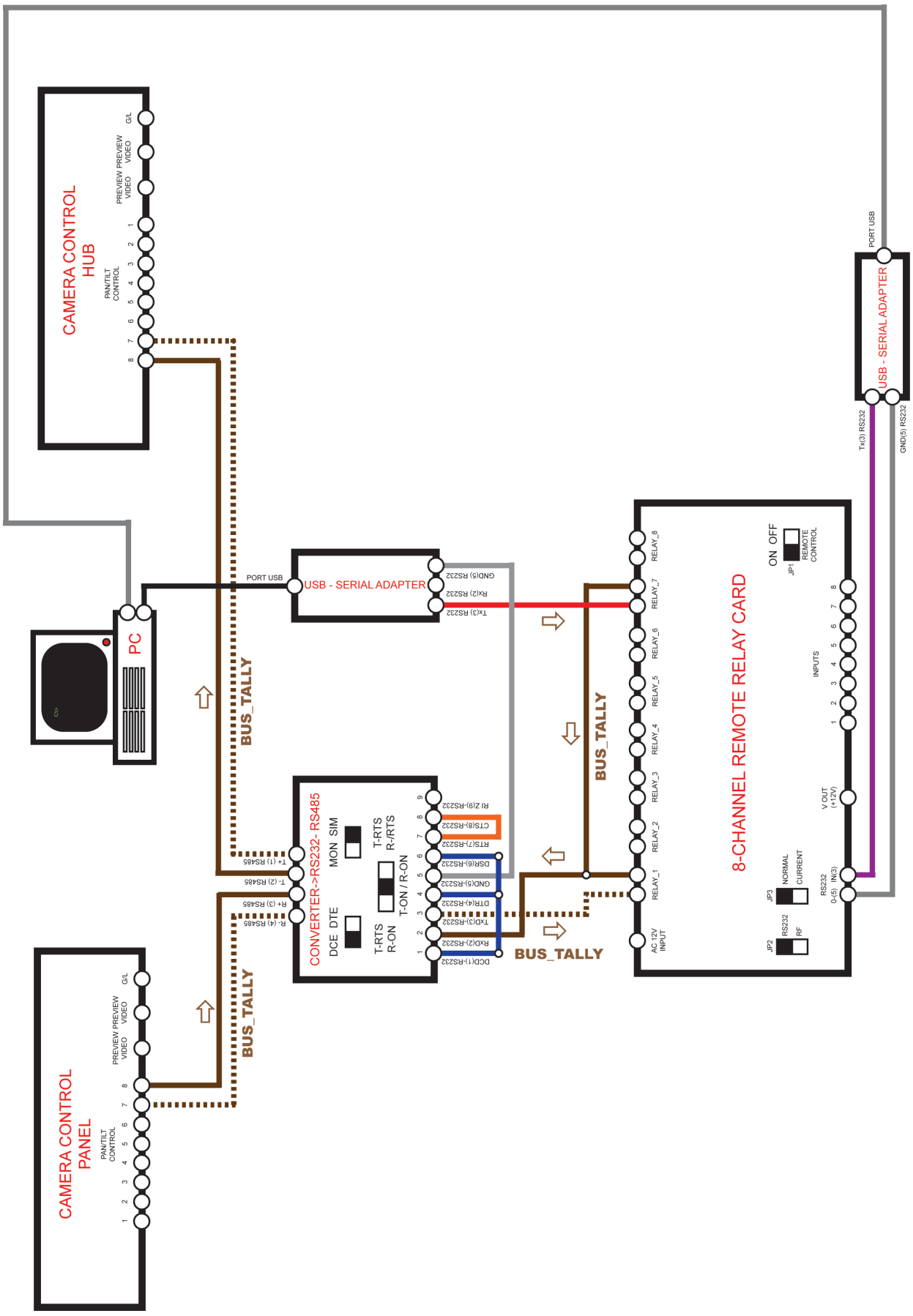
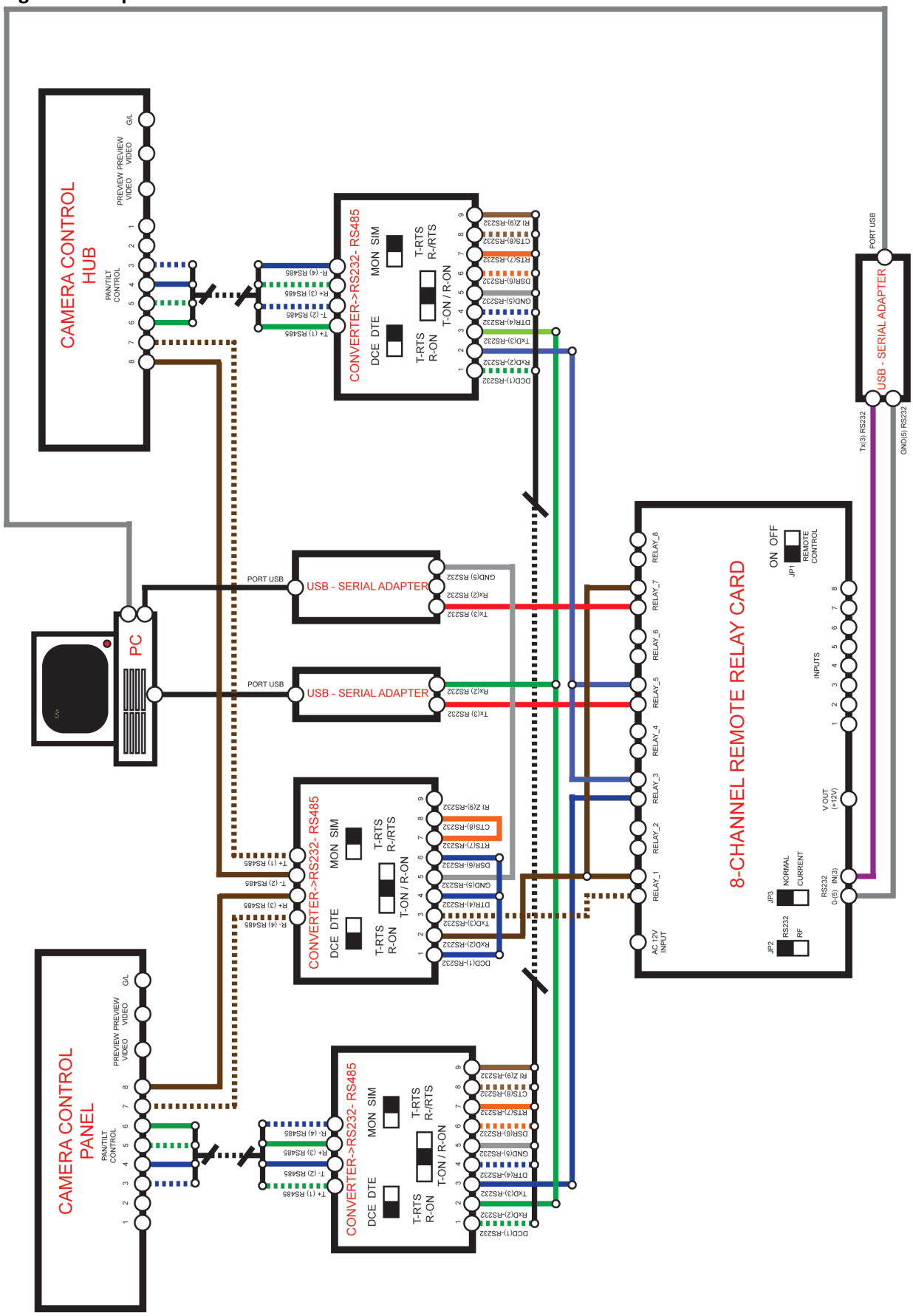


Diagrama Completo de conexiones:



5.2. Interface RS232C: De la Cabeza P/T al PC

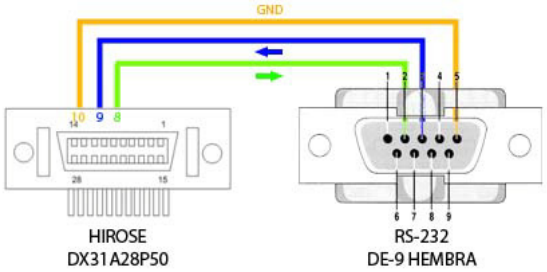
Este interface trata de un cable que permite conectar la cabeza P/T con un PC, a través del conector "P/T CONTROL IN (CONTACT/RS-232C)", utilizando "el Protocolo de Comunicación RS-232C".

El interface permite un número ilimitado de cámaras conectadas al PC, ya que el PC esta directamente conectado al cabezal P/T por medio de un puerto serie para cada cámara.

Una ventaja es que el Panel y el PC siempre están habilitados, porque en el caso de que ambos envíen un comando en el mismo instante, tendrá prioridad la operación de control iniciada desde el panel (AW-RP505).

El único inconveniente es la adición de más cableado en el sistema de cámaras y en el plató de grabación.

La configuración del cable "AW-CA28T9" es en Null-Modem (Modem Nulo). Es un método para conectar dos terminales usando un cable serie. En la confección null módem las líneas de transmisión y recepción están cruzadas.



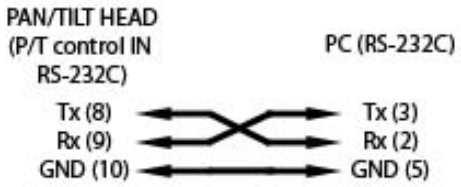


Fig. 1 y Fig.2: "Diagrama de conexión del cable AW-CA28T9"

El cable deberá tener un área de sección transversal de 0,3 mm^2 como mínimo y que cumpla con las disposiciones de la "Ley de Control de Materiales y Aparatos Electrónicos". La longitud máxima será de 15 metros según marca la norma RS-232C.

En la siguiente tabla mostramos la configuración del conector "HIROSE DX31A28P50" o "HIROSE DX30-28P (50)", ambos conectores son iguales la única diferencia es que cada uno es para un tipo de cable diferente:

Configuración Conector HIROSE			
Nº de Patilla	**Señal de Contacto**	**Señal de RS-232C**	**Funciones**
8	TX	TXD	Datos de Tx de la cabeza Pan/Tilt.
9	RX	RXD	Datos de Rx de la cabeza Pan/Tilt.
10	GND	GND	----------

Conseguir este tipo de conector de HIROSE es bastante difícil, ya que no se puede adquirir en cualquier tienda de electrónica. Pero hay una empresa Francesa "MOUSER ELECTRONICS" que es una de las distribuidoras de HIROSE en Europa, que si que poseen el conector y lo puedes adquirir on-line:

http://nl.mouser.com/ProductDetail/Hirose-Electric/DX30-28P50/?qs=XQjbzJWzFPWREbXPR8DrzQ%3d%3d
http://fr.mouser.com/

Y aquí tenemos el catalogo de los conectores HIROSE:

..\Ficheros anexos\Documentación\DX SERIES HIROSE.pdf

En esta segunda tabla mostramos la configuración de pines del conector RS-232C del PC:

Configuración Conector RS-232		
Nº de Pin	**Señal de Rs-232**	**Descripción**
1	---	Masa chasis
2	Rx	Recepción de Datos
3	Tx	Transmisión de Datos
4	DTR	Terminal de Datos Lista
5	GND	Señal de Tierra
6	DSR	Equipo de Datos Listo
7	RTS	Solicitud de Envío
8	CTS	Borrar para Enviar
9	RI	Indicador de Llamada

Por lo tanto el diagrama de conexiones del sistema de cámaras para una distancia menor de 15m, quedará de la siguiente forma:

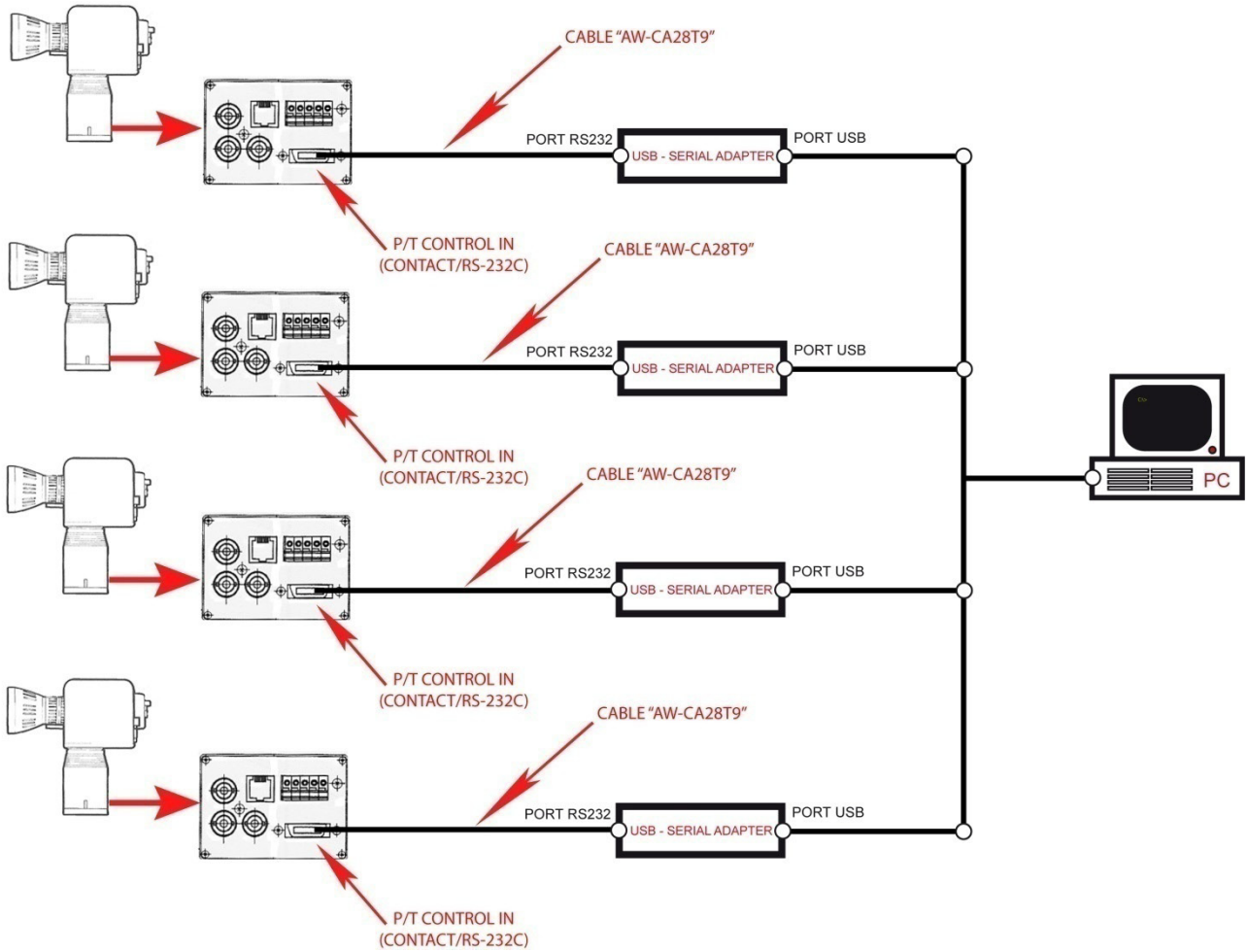
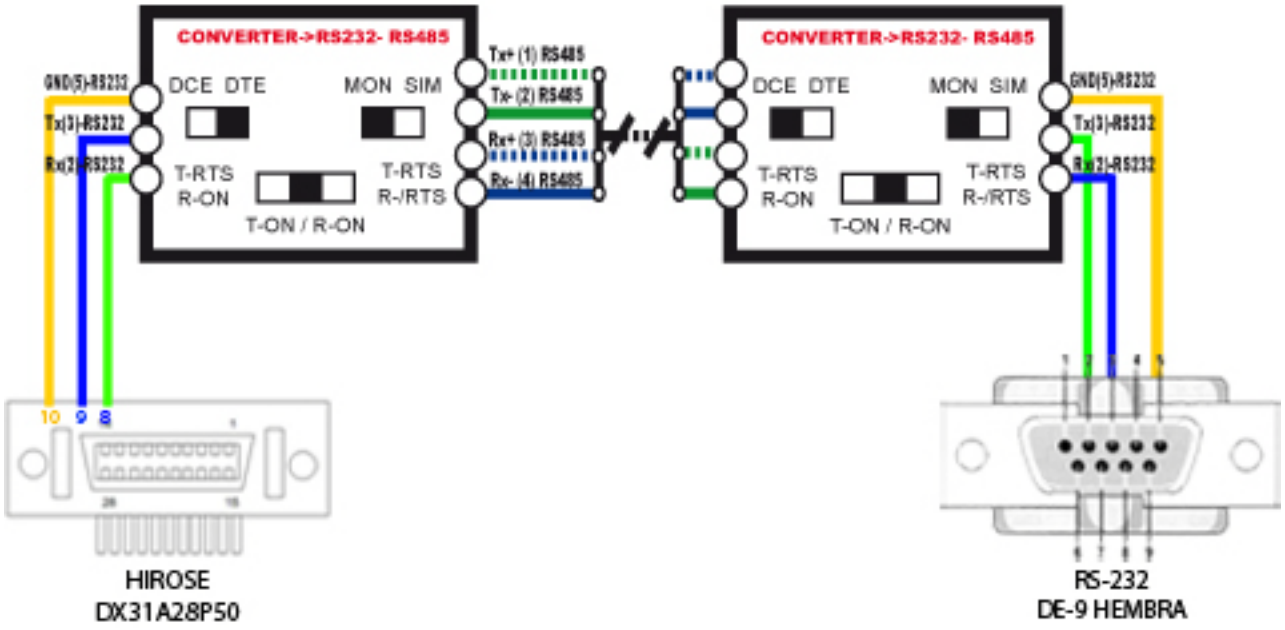


Fig. 3: "Diagrama de conexiones del sistema de cámaras para distancias menores de 15m"

Si queremos extender más la longitud del cable, deberemos utilizar dos conversores RS-232C a RS-422, uno para cada extremo del cable. Este montaje nos permitirá obtener longitudes de hasta 500 metros.



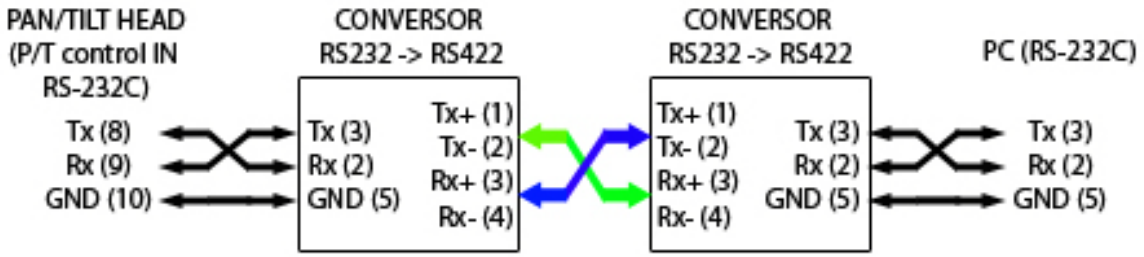


Fig. 4 y Fig. 5: "Diagramas de conexiones del cable AW-CA28T9 para distancias menores de 500m"

Y el diagrama de conexiones del sistema de cámaras para una distancia menor de 500m quedará de la siguiente forma:

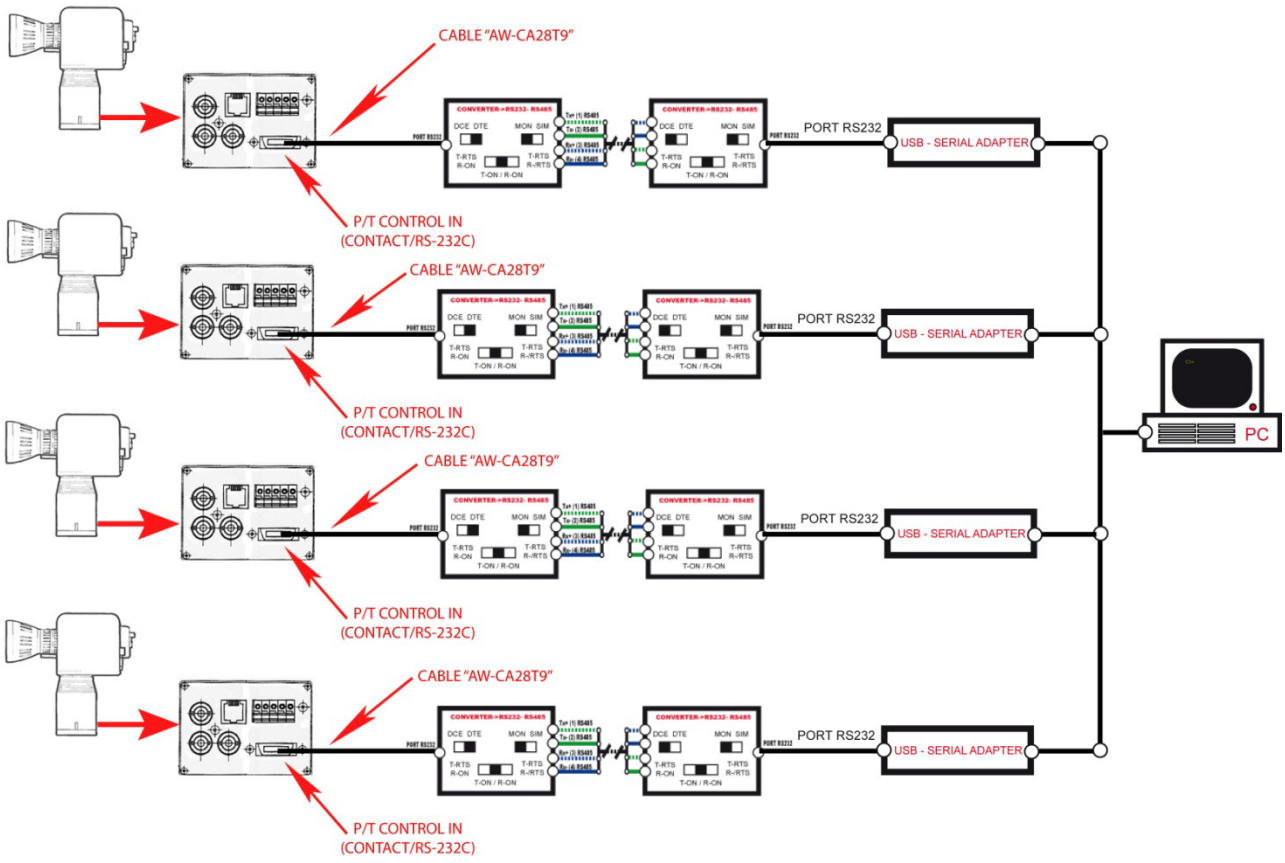


Fig. 6: "Diagrama de conexiones del sistema de cámaras para distancias menores de 500m"

Los archivos de los diagramas están incluidos en el DVD, en la carpeta:

Ver: ..\Ficheros anexos\DIAGRAMAS

5.3. Comparación entre Interfaces

Una de las principales diferencias entre estos interfaces es el número de cámaras que podemos llegar a controlar.

En el interface RS-422 el número de cámaras viene limitado por las características del Hub, en este solo se pueden llegar a controlar un máximo de 5 cámaras. En cambio, en el interface RS-232C el número de cámaras que podemos llegar a conectar es ilimitado, ya que cada cámara va conectada a un puerto serie diferente.

Otra diferencia es que el protocolo RS232C necesita la incorporación de cableado, un cable para cada cámara. En cambio, en el interface RS-422 nos conectamos al cable 'P/T CONTROL' que conecta el panel y el Hub. Pero al conectar el PC tenemos que desconectar el panel para que nos se produzcan interferencias, por lo tanto deshabilitamos el panel cada vez que usemos el PC.

Para el interface RS-422 necesitaremos tres conversores RS232 a RS-422, tres puertos serie y una tarjeta de control remoto de relés. En cambio, para el interface RS-232C no necesitaremos de ningún conversor solamente de un puerto serie por cada cámara si la distancia es menor de 15m. Y Si la distancia es mayor de 15m deberemos colocar 2 conversores RS232 a RS422 por conexión, esto nos permitirá alcanzar hasta distancias de 500m.

6. Software

El entorno de desarrollo que vamos a utilizar para programar las aplicaciones es Visual C++ (también conocido como MSVC, Microsoft Visual C++). Esta especialmente diseñado para el desarrollo y depuración de código escrito para las API's de Microsoft Windows, DirectX y la tecnología Microsoft .NET Framework.

Visual C++ hace uso extensivo del framework Microsoft Foundation Classes (o simplemente MFC), el cual es un conjunto de clases C++ para el desarrollo de aplicaciones en Windows.

Nosotros desarrollaremos nuestras aplicaciones en un proyecto de Windows Forms utilizando Visual C + +.

Desarrollaremos dos software, uno para cada protocolo y de esta manera comprobaremos: qué protocolo es el adecuado, cuáles son sus ventajas e inconvenientes y sus funciones. De los dos software elegiremos uno, el más adecuado, flexible y que funcione correctamente.

El software elegido será utilizado en el plató de televisión como segundo panel, por lo tanto desarrollaremos más su funcionalidad con el diseño de métodos, formularios 'Forms' (configuración, DialogBox…), menú (ToolStripMenuItem), dll's (para el Joystick)… Todo lo que sea necesario para que el programa sea funcional y viable. De todo esto lo hablaremos en los siguientes apartados, además de explicar con más detalle los métodos y funciones del Software elegido.

6.1. Software para RS-422

Este programa está basado en la simulación de la comunicación que establece el panel de control con el Hub, es decir en el protocolo de comunicación RS-422. Tal y como hemos explicado en el apartado "4.1. Protocolo de comunicación RS422: Del Panel de control [AW- RP505] al Hub [AW-HB505]", la comunicación se realiza a través de 3 puertos por lo tanto necesitamos declarar 3 puertos serie por los cuales enviaremos y recibiremos las secuencias de comandos a través de los buses.

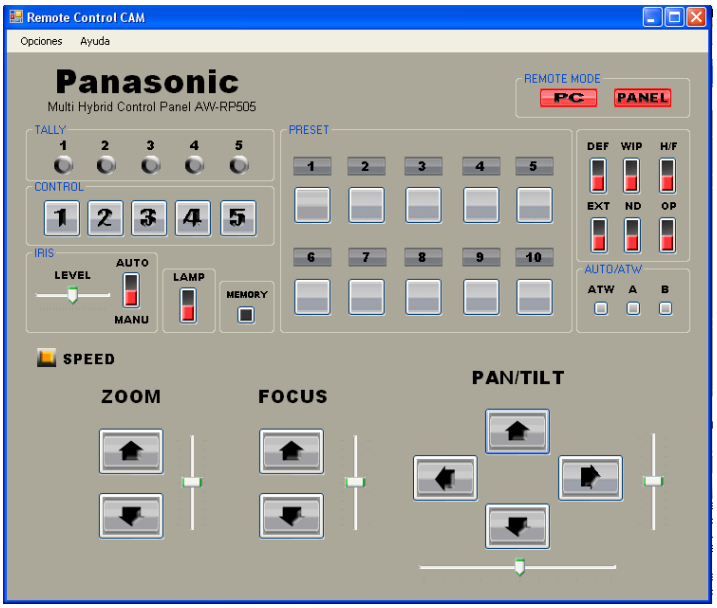


Fig. 1: "Imagen del software 'controlCAM_c++_ToHub_RS422' "

Para sincronizar la comunicación entre el Hub y el Pc cada vez que recibimos el comando de Command Status “I50505050554770..” desde el BUS_TX_P/T del panel, se ejecuta el evento de ‘DataReceived()’del puerto serie conectado al BUS_TX_P/T,. En el cual está introducido el código para la ejecución de la secuencia de comandos correspondiente, dependiendo del control del panel que ha sido pulsado (Cambio de cámara, seleccionar un Pre-set, etc.).

Hemos elegido el comando ‘status command’, en vez del comando de resultado ‘E0.H0.’ proveniente del Hub, para sincronizar el PC con el Hub. Porque el comando ‘status command’ es el de menor intervalo de tiempo 47ms, en cambio el comando de resultado del Hub es de 500ms y esto puede introducir interrupciones en la comunicación, si el puerto serie no recibe el comando de resultado en ese mismo intervalo. Cosa que al elegir el comando ‘status command’ cuando suceda esta la interrupción sería en un intervalo menor de tiempo y no se produciría errores ni cortes prolongados en la comunicación entre el PC y el Hub.

<Código C++>

```
private: System::Void serialPort_BUS_MEM_DataReceived(System::Object^  sender,
System::IO::Ports::SerialDataReceivedEventArgs^  e) {
      //Metodo utilizado para recibir los comandos de resultados del BUS_MEM_RX
y contestar el cambio de camara

      array<unsigned char>^ bufferEntradaMEM_RX= gcnew array<unsigned char>(17);

      serialPort_BUS_MEM->Read(bufferEntradaMEM_RX,0,17);

      switch(bufferEntradaMEM_RX[0]){

            case 0x49: //Comando de estado del Panel ASCII -> 'I'

                  if(flag_Evento_MEM_StatusCommand==true){

                        //Ejecutamos Evento_MEM_HUB_StatusCommand()
                        Evento_MEM_StatusCommand(sender,e);

                  }//if()
                  if(flag_EventoCambioCAM==true){

                       //Ejecutamos EventoCambioCam()
                       EventoCambioCam(sender,e);

                       //inicializamos flag_CAM
                       flag_EventoCambioCAM=false;

                  }//if()
                  if(flag_EventoCambioMEM==true){

                        //Ejecutamos EventoCambioCam()

                        EventoCambioMem(sender,e);

                        //inicializamos flag_CAM
                       flag_EventoCambioMEM=false;

                  }//if()

                  if(flag_EventoCambioWhiteBalance==true){

                        //Ejecutamos EventoCambioCam()

                        EventoCambioWhiteBalance(sender,e);

                        //inicializamos EventoCambioWhiteBalance
                       flag_EventoCambioWhiteBalance=false;
```

```
                    }//if()

                    break;

              };//switch()

}
```

<End>

Los otros dos puertos serie están conectados a:

- Uno al BUS_TALLY, por el cual nuestro programa envía el comando de la cámara seleccionada (excepto en la cámara 1 que no se envía el comando TALLY) a la vez que enviamos los comandos del BUS_TX_PT. Como podemos observar en el código siguiente del Evento 'Evento_MEM_StatusCommand()':

<Código C++>

```
public: System::Void Evento_MEM_StatusCommand(System::Object^  sender,
System::EventArgs^  e) {

      switch(flag_SerialPortCAM){
            case true:

                  //envio de valor de camara
                  serialPort_BUS_CAM->Write(bufferSalidaCam_TallyCommand,0,1);


                  //envio de valor de StatusCommand -> I50505050554770..
                  serialPort_BUS_MEM->Write(bufferSalidaStatusCommand,0,17);

                  break;

            case false:


                  //envio de valor de StatusCommand -> I50505050554770..
                  serialPort_BUS_MEM->Write(bufferSalidaStatusCommand,0,17);

                  break;
       };//switch()
                        }
```

<End>

- Y el otro está conectado a la tarjeta de control remoto de relés, que conmutará la conexión entre el Hub y el panel o el PC. En el programa este evento se realizará por medio de los botones 'PC' y 'PANEL' de "REMOTE MODE".

<Código C++>

```
private: System::Void PANEL_Click(System::Object^  sender, System::EventArgs^
e) {

       //Apagar luces botones remote mode
            PANEL->ImageIndex=0;
            REMOTE_PC->ImageIndex=0;

      //Encender luces boton Panel remote mode
            PANEL->ImageIndex=1;

      rele.Activar_Panel(gcnew array<unsigned char>(5),serialPort_BUS_RELES);
```

```
            //retado activacion desactivacion reles
             Thread::CurrentThread->Sleep(375);

             //desactivar flag_Evento_MEM_HUB_StatusCommand
            flag_Evento_MEM_StatusCommand=false;
             }
private: System::Void REMOTE_PC_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Apagar luces botones remote mode
            PANEL->ImageIndex=0;
            REMOTE_PC->ImageIndex=0;

      //Encender luces boton Panel remote mode
            REMOTE_PC->ImageIndex=1;

rele.Activar_Bus_Mem_Cam_Pc(gcnew array<unsigned char>(5),serialPort_BUS_RELES);

            //Activar flag_Evento_MEM_HUB_StatusCommand
            flag_Evento_MEM_StatusCommand=true;

 }
```

<End>

Y Finalmente cada vez que pulsamos un botón de una función del software como: seleccionar un Pre-set, una cámara o un balance de blancos. Se produce una interrupción en la comunicación que envía la secuencia de comandos correspondiente para cada función. Como podemos observar con los códigos siguientes para seleccionar un cambio de cámara (seleccionamos la cámara 2):

El primer fragmento de código muestra el evento relacionado con el botón de selección de la cámara 2, donde activamos el 'flag_EventoCambioCAM' para que se envíe la secuencia de comandos por medio del evento serialPort_BUS_MEM_DataReceived y introducimos los valores de los comandos de la cámara seleccionada en las variables para el BUS_TALLY: 'bufferSalidaCam_Command_1', 'bufferSalidaCam_Command_2' y 'bufferSalidaCam_TallyCommand'.

<Código C++>

```
private: System::Void Evento_Cam2(){
       //Reseteamos la luz de Tally
       Evento_ResetTally();

       //Encender luz de tally_2
       TALLY_2->ImageIndex = 1;

       //valor de camara_2 command_1 -> .. -> 0x00
       bufferSalidaCam_Command_1[0]=0x00;

       //valor de cambio camara 2 ->7E 7C
       bufferSalidaCam_Command_2[0]=0x7E;
       bufferSalidaCam_Command_2[1]=0x7C;

       //Valor bufferSalidaCam_TallyCommand para la camara 2-> | -> 0x7C
       bufferSalidaCam_TallyCommand[0]=0x7C;


      //DesHabilitamos flag_CAM_1
       flag_CAM_1=false;

      //DesHabilitamos el evento MEM_HUB_StatusCommand
       flag_Evento_MEM_StatusCommand=false;

       //Habilitamos el evento cambio de camara
       flag_EventoCambioCAM=true;
}
```

<End>

Y en este segundo fragmento de código mostramos la secuencia de comandos para el cambio de cualquier cámara que enviamos a través de los puertos series conectados a los Buses: BUS_TX_P/T y BUS_TALLY.

<Código C++>

```
public: System::Void EventoCambioCam(System::Object^  sender, System::EventArgs^
e) {

      //Comandos para el envio del cambio de camara

       if(flag_SerialPortCAM==true){
            //envio de valor de camara
      serialPort_BUS_CAM->Write(bufferSalidaCam_TallyCommand_Anterior,0,1);
      }//if()

      //envio de valor de peticion de cambio camara -> Z****.
       serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_1,0,6);


       //retado de envio de datos por puerto
       Thread::CurrentThread->Sleep(16);
       //envio de valor de Status_command -> I50505050554770.
       serialPort_BUS_MEM->Write(bufferSalidaStatusCommand,0,17);

       //retado de envio de datos por puerto
       Thread::CurrentThread->Sleep(188);

      //Envio_1 de X0****. X0****. y 00.

      //envio de valores de ChangeCam_Command_2 -> X0****.X0****.
      serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_2,0,7);
      serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_2,0,7);

      if(flag_CAM_1==false){
            //envio de valor de camara Command_1
            serialPort_BUS_CAM->Write(bufferSalidaCam_Command_1,0,1);
      }//if()

      //retado de envio de datos por puerto
       Thread::CurrentThread->Sleep(40);

      //Envio_2 de X0****. X0****. y 00.


      //envio de valores de ChangeCam_Command_2 -> X0****.X0****.
      serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_2,0,7);
      serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_2,0,7);

      if(flag_CAM_1==false){
            //envio de valor de camara Command_1
            serialPort_BUS_CAM->Write(bufferSalidaCam_Command_1,0,1);
      }//if()

      //retado de envio de datos por puerto
      Thread::CurrentThread->Sleep(16);

      //Envio de S****.

      //envio de valor de comando para la informacion de memoria Pre-set ->
S****.
      serialPort_BUS_MEM->Write(bufferSalidaChangeCam_Command_3,0,6);

      if(flag_CAM_1==false){
            //envio de valor de camara Command_2
            serialPort_BUS_CAM->Write(bufferSalidaCam_Command_2,0,2);
      }//if()
```

```
      //retado de envio de datos por puerto
       Thread::CurrentThread->Sleep(16);

       //Establecemos el flag SerialPortCAM si estamos en la cam1 o no.
       if(flag_CAM_1==true){

             //Deshabilitamos flag SerialPortCAM
             flag_SerialPortCAM=false;
       }//if()

       if(flag_CAM_1==false){
             //Habilitamos flag SerialPortCAM
             flag_SerialPortCAM=true;
       }//if()

       //Habilitamos el evento MEM_HUB_StatusCommand
       flag_Evento_MEM_StatusCommand=true;

       //Guardamos el valor de bufferSalidaCam_TallyCommand a
bufferSalidaCam_TallyCommand_Anterior
       bufferSalidaCam_TallyCommand_Anterior[0]
=bufferSalidaCam_TallyCommand[0];

      //Apagar luces botones remote mode
      PANEL->ImageIndex=0;
      REMOTE_PC->ImageIndex=0;

      //Encender luces boton Panel remote mode
      REMOTE_PC->ImageIndex=1;

                  }
```

<End>

Pero para la ejecución de las funciones de Pan/Tilt, Zoom, focus e iris, además de los interruptores de condición están introducidas en el comando 'Status command' que se envía cada 47 ms, cuando recibimos es mismo comando por el panel a través del evento 'serialPort_BUS_MEM_ DataReceived ()', como hemos explicado anteriormente en este apartado.

En los siguientes fragmentos de código mostramos un ejemplo de cuando pulsamos el botón 'Right' de la función Pan, que ejecuta un evento el cual introduce el valor de sensibilidad Pan del `trackBar_sensibilidadPan` en el coeficiente 1 y 2 del array 'bufferSalidaStatusCommand' correspondientes al comando Pan, tal y como hemos explicado en el protocolo de comunicación RS-422 en el apartado "4.1.2. Comando para el Joystick y el panel de información de interruptores (Status Command)".

El primer fragmento de código muestra cuando pulsamos y mantenemos presionado el botón:

<Código C++>

```
private: System::Void Evento_RightDown(){

       //Encender luz de arrow_RIGHT
      RIGHT->ImageIndex=1;

       int sensibilidad;
       int DEC_decimal=0;
       int DEC_unidad=0;

      sensibilidad=trackBar1->Value;
      sensibilidad=50-sensibilidad;
```

```
      DEC_decimal=sensibilidad/10;
      DEC_unidad=sensibilidad-DEC_decimal*10;

      //valor de MCP tilt de camara -> I50505050554770.. ->Pan left
      bufferSalidaStatusCommand[1]=0x30+(unsigned char)abs(DEC_decimal);
      bufferSalidaStatusCommand[2]=0x30+(unsigned char)abs(DEC_unidad);

      //Calculo del valor CheckSum
      bufferSalidaStatusCommand[15]=NULL;

      for(int i=0;i<15;i++){

      bufferSalidaStatusCommand[15]=bufferSalidaStatusCommand[15]+bufferSalidaSt
atusCommand[i];
      }//for()
}
```

<End>

Y en este segundo fragmento de código se produce cuando se suelta el botón y se introduce el valor de parada de movimiento Pan:

<Código C++>

```
private: System::Void Evento_LeftUp(){
      //Apagar luz de arrow_LEFT
      LEFT->ImageIndex=0;

      bufferSalidaStatusCommand[1]=0x35;  //inicializamos valor Pan_LEFT joystick
      bufferSalidaStatusCommand[2]=0x30;

      //Calculo del valor CheckSum
      bufferSalidaStatusCommand[15]=NULL;

      for(int i=0;i<15;i++){

      bufferSalidaStatusCommand[15]=bufferSalidaStatusCommand[15]+bufferSalidaSt
atusCommand[i];
      }//for()

 }
```

<End>

Las demás funciones de movimiento: Tilt, Zoom, Focus e Iris funcionan de la misma forma, únicamente cambia el lugar de los coeficientes del array 'bufferSalidaStatusCommand', por ejemplo para la función Tilt introduciremos los valores de sensibilidad de movimiento en los coeficientes 3 y 4.

Y en los interruptores de condición únicamente cambiaremos su estado, introduciendo el valor de estado de los mismos en los coeficientes del array 'bufferSalidaStatusCommand': 11, 12, 13 y 14.

En el siguiente código mostramos un ejemplo de cuando pulsamos el interruptor 'DEF', que lo activa o desactiva según su estado anterior e introduce el valor en el coeficiente correspondiente del array 'bufferSalidaStatusCommand':

```
private: System::Void DEF_Click(System::Object^  sender, System::EventArgs^  e)
{
      //Comando del interruptor del deshelador DEF -> ON/OFF
       switch(SW_StatusBits_2[0]){
            case '0':

                  //Valor bit de activacion DEF ->modo OFF
                  SW_StatusBits_2[0]='1';
                  //Habilitamos el boton DEF
```

```
                    DEF->ImageIndex=0;
                     break;
               case '1':

                     //valor bit desactivacion DEF -> modo ON
                     SW_StatusBits_2[0]='0';
                    //Habilitamos el boton DEF
                     DEF->ImageIndex=1;
                     break;

        };//switch()

       //Convertimos los bits de informacion SW_StatusBits_2 de binario a decimal
        double SW_Status_2=0.0;
        for(int i=0;i<4;i++){
              if(SW_StatusBits_2[i]=='1'){
                    SW_Status_2=SW_Status_2+pow(2.0,i);
              }//if()
        }//for()
       //Lo  guardamos en el Byte 12 de informacion de StatusCommand: DEF_SW(bit
0),WIPE_SW(bit 1),HEAT_SW(bit 2) y CERO(bit 3)
       bufferSalidaStatusCommand[12]=0x30+(unsigned char)abs(SW_Status_2);

       //Calculo del valor CheckSum
       bufferSalidaStatusCommand[15]=NULL;

       for(int i=0;i<15;i++){

       bufferSalidaStatusCommand[15]=bufferSalidaStatusCommand[15]+bufferSalidaSt
atusCommand[i];
       }//for()

}
```

<End>

Hemos descartado este software porque al probarlo solo se puede controlar la cámara 1, cuando realizamos un cambio de cámara a otra cámara que requiera el comando TALLY (como la 2, 3, 4 y 5) se produce un error en la comunicación y no cambia de cámara, sino se queda en la cámara 1 y se producen interferencias en la cámara 1 causadas por el comando TALLY que entorpecen el movimiento.

Esto es consecuencia de un error de sincronización del envío del comando de Tally con el resto de las secuencias de comandos enviadas a través de los buses (BUS _TX_P/T, BUS_RX_P/T), ya que los puertos series a velocidades menores de 100 ms no son tan precisos al enviar los comandos.

También es provocado por la “Señal de control de cámara” enviada por el panel a través del conector “Camera Control IN/OUT” que valida el comando de TALLY por medio de una secuencia sincronizada de 47 ms con el comando de Tally formada por el valor ‘00’. Este problema no sucede con la cámara 1 porque el comando de Tally para esa cámara no se envía y en este caso funciona perfectamente.

El código completo está contenido en la carpeta del proyecto:
..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHub_RS422

6.2. Software para RS-232

El desarrollo de este software está basado en el protocolo de comunicación RS-232. Es mucho más sencillo de programar, porque sus comandos actúan como eventos y no se envía contantemente una secuencia de comandos sincronizada, tal y como sucede en el Protocolo de comunicación RS-422.

También posee más funciones que el panel y el protocolo RS-422 como:

- 50 Pre-sets, que con el Protocolo RS-422 y el panel únicamente poseemos 10 Pre-Sets.

- La posibilidad de establecer una posición exacta Pan/Tilt y de Lentes en grados (sólo para cabezales AW-PH360L). Esto nos permitirá poder saber la posición de las cámaras en todo momento y desarrollar funciones para cargar/guardar la posición actual de cada cámara además de monitorizar gráficamente la posición de la cabeza P/T y de las lentes.

Como solo podemos utilizar esta última función para los cabezales P/T "AW-PH360L" desarrollaremos dos versiones del software, uno para ambos modelos (AW-PH300A/AW-PH360L) y otra versión solo para el modelo "AW-PH360L".

¡¡¡Atención!!!: La versión de Software para el modelo "AW-PH300A" también es válida para el modelo "AW_PH360L", pero no a la inversa.

Ambas versiones son muy parecidas, ya que poseen muchas funciones en común del protocolo de comunicación. Únicamente difieren en:

- Los controles de posición del Pan/Tilt y Lentes.
- La monitorización y establecimiento de la posición de las cámaras.
- La Carga y el Save de las posiciones del cabezal P/T y las lentes en un archivo txt.

Como podemos observar en las siguientes imágenes de la interfaz grafica del Software hay algunas diferencias entre versiones.

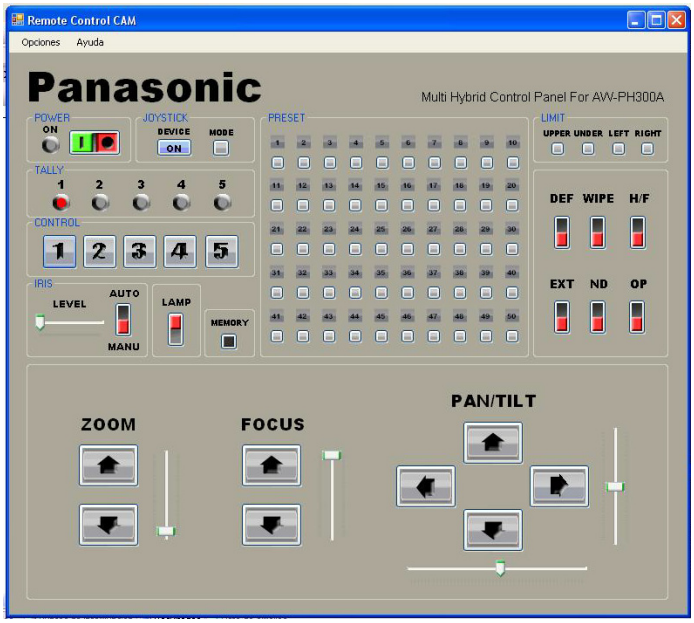


Fig. 2: "Imagen del software Versión para los modelos AWPH300A/AW-PH360L"

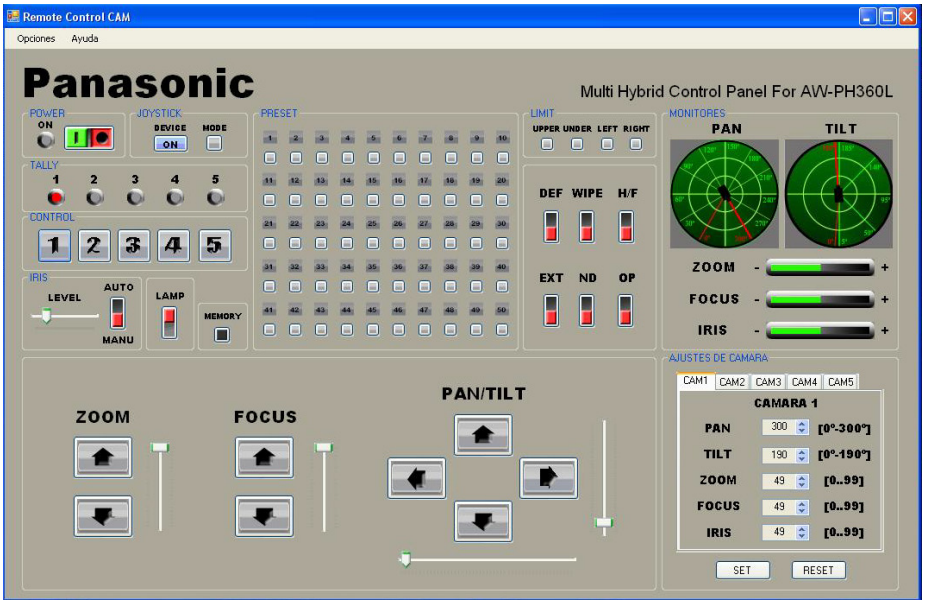


Fig. 3: "Imagen del software Versión solo para los modelos AW-PH360L"

Para ambas versiones integraremos el dispositivo "Joystick", para poder controlar de mejor forma los movimientos P/T y Lentes de cámara. Desarrollaremos dos modos de sensibilidad o velocidad de movimiento: Uno con sensibilidad variable, tal y como sucede en el Panel de Control, y otro con sensibilidad fija que estableceremos por medio de los TrackBars de la interfaz Gráfica.

Este último modo permitirá realizar a nivel artístico planos en movimiento de velocidad constante, cosa que con el panel de control no sucede ya que los movimientos de las palancas P/T y Lentes tienen una velocidad variable y poca sensibilidad. Es obvio que esta cualidad a nivel de producción es un gran avance.

El único inconveniente es a nivel de interface, ya que hay que cablear una conexión diferente para cada cámara, desde el PC hasta la cabeza P/T.

Pero también es una ventaja no tener que utilizar una tarjeta remota de relés para conmutar entre el panel y el PC. Esto provocaba la desconexión temporal del panel y por lo tanto la inhabilitación del mismo. Con el protocolo RS-232, tanto el PC como el panel están habilitados constantemente.

Al probar ambas versiones del software RS232C en el sistema de cámara hemos comprobado que funcionan perfectamente y no se produce ningún problema de comunicación, además de que podemos controlar todas las cámaras si ningún problema. Cosa que con el software RS-422 no ocurre, ya que como hemos explicado no nos permite seleccionar otras cámaras, únicamente la cámara 1.

Y por lo tanto, hemos desarrollado más estas versiones de Software RS-232C introduciendo mejoras como: guardar el estado de los interruptores de condición, el valor de las sensibilidades y las posiciones de cada operador de posición P/T y lentes (esto último únicamente para el modelo de cabeza P/T AW-PH360L). Además integraremos un dispositivo como el joystick y diseñaremos para la versión AW-PH360L unos monitores de posición Pan/Tilt y lentes.

Si se desea desarrollar más el Software los archivos del proyecto Windows Forms, la cabecera de los comandos de control de cámara (“CamCommands”) está incluida en el contenido del Dvd:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH300A

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L

..\Ficheros anexos\Software y cabeceras\Cabeceras\CamCommands.h

6.2.1. Encabezados y espacios de nombre:

Deberemos de incluir las siguientes librerías y espacios de nombres:

<Código C++>

```
using namespace System;
using namespace System::Globalization;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Collections::Generic;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Text;
using namespace System::IO::Ports;
using namespace System::Threading;
using namespace System::Diagnostics;

#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "string.h"

//Definimos la clase que controla las camaras
CamCommands cam_commands;
```

<End>

6.2.2. Puerto serie y cambio de cámara

Cada cámara irá enchufada a un puerto serie, tal y como hemos explicado anteriormente en el apartado (5.2. Interface RS-232C: De la Cabeza P/T al PC) .

Para mantener el puerto abierto mientras se ejecute un envío o una recepción de datos por el puerto serie, tendremos que escribir el siguiente código dentro del constructor del Formulario principal:

<Código C++>

```
 // Abrir puerto de CAM mientra se ejecute la aplicación
if (!serialPort_BUS_CAM->IsOpen)
      {
      try{
            serialPort_BUS_CAM->Open();
      }catch(Exception ^ex){
            MessageBox::Show(ex->ToString());
      }
}
```

<End>

En nuestro programa en vez de declarar varios puertos, declararemos únicamente uno. Y el cambio de cámara lo realizaremos cambiando la propiedad “PortName” de la clase “System::IO::Ports::SerialPort”.

Además de utilizar varios métodos para guardar y cargar el estado actual de cada cámara. Y establecer la última posición de: Pan/Tilt, lentes (Zoom, Focus y Iris), límite, condición de los interruptores y los TrackBars de sensibilidad P/T de la cámara seleccionada.

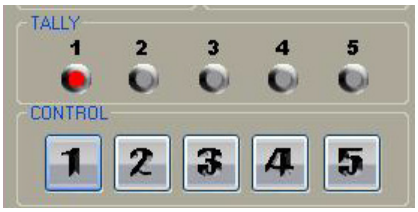


Fig. 4: “Imagen del selector de cámara”

El código completo de todos los cambios de cámara esta incluido en el archivo de código “Form1.h” del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
//************Comandos para el cambio de ************************************
public: System::Void  Evento_Cam1(){

      //Reseteamos la luz de Tally
       Evento_ResetTally();

      //Encender luz de tally_1
       TALLY_1->ImageIndex = 1;

      //Guardamos el valor de la posicion pan/tilt de la ultima camara
      seleccionada
      SavePanTiltPosition();

      //Guardamos el valor de la posicion LensCondition de la ultima camara
      seleccionada
      SaveLensCondition();

      //Guardamos las VariablesConditionControl
      SaveVariablesStatusLimit();

      //Establecemos el valor numCamSelected
      numCamSelected=1;

      //Selecciona la pagina de tabControlAjustes para la camara correspondiente
       tabControlAjustes->SelectedIndex=numCamSelected-1;

      //cerramos puertos
      this->serialPort_BUS_CAM->Close();

      //Cambiamos el nombre de los puertos
      this->serialPort_BUS_CAM->PortName = NombreCom_Cam1;

      //Abrimos puertos
      this->serialPort_BUS_CAM->Open();

       //Cargamos las variables de control de condicion
       LoadVariablesConditionControl();

      //Cargamos las variables de StatusLimit
```

```
        LoadVariablesStatusLimit();

        //Encendemos el ultimo preset seleccionado
        LoadUltimaSeleccionPreset();


        //Cargamos los valores actuales TrackBarPosition para la camara
        selecionada
        LoadTrackBarPosition();

        //Cargamos la posicion Pan Tilt de la camara correspondiente
        LoadPanTiltPosition();

        //retado de envio de datos por puerto
        Thread::CurrentThread->Sleep(100);

        //Cargamos la posicion LensCondition de la camara correspondiente
        LoadLensCondition();

        //Establecemos la posicion actual de los monitores PAN/TILT
        Evento_Establecer_Monitor_PanTiltPosition();

        //Establecemos la posicion actual de los monitores LensCondition
        Evento_Establecer_Monitor_LensCondition();

}
```

<End>

6.2.3. Controles Pan/Tilt

Esta función es idéntica para ambas versiones. Porque al utilizar la función 'Evento_PanTiltCommand()' repetidas veces, en la versión del modelo AW-360L, se producen movimientos torpes e discontinuos.

Pero podemos calcular la nueva posición para cargarla en los monitores P/T. Ya que cuando encendemos las cámaras o cambiamos de cámara se introduce el valor de la última posición tanto en la cabeza P/T como en el programa. En cambio esto no vale para los modelos AW-300A porque no contienen el comando 'Evento_PanTiltCommand()'.

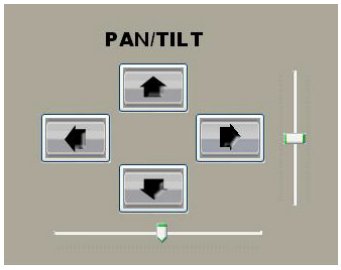


Fig. 5: "Imagen de los comandos de movimiento Pan/Tilt"

Por lo tanto para estos controles utilizaremos los comandos de operación Pan/Tilt del protocolo de comunicación RS-232 (mirar los apartados 4.2.2 y 4.2.3). Estos comandos no devuelven ni establecen un posición determinada Pan/Tilt, simplemente mueven la cabeza de cámara. Además funcionan en ambos modelos de cabeza de cámara.

Los comandos los introduciremos en un Timer 'timer_PanTiltCommand_Tick() ' con un intervalo de tiempo de 100ms. Y los ejecutaremos mientras uno de los controles P/T este pulsado. Y cuando soltemos el control ejecutara el comando ''con el valor de parada una sola vez.

Estos comandos los enviamos varias veces cuando el botón está presionado, porque si está habilitado el botón de modo sensibilidad variable del joystick 'ModeSens', el cabezal P/T pueda variar la velocidad de movimiento según el movimiento de la palanca, si está deshabilitado la velocidad de movimiento P/T será constante y estará establecida por los TrackBars de sensibilidad.

Fig. 6: "Imagen del control del Joystick"

<Código C++>

```
private: System::Void timer_PanTiltCommand_Tick(System::Object^  sender,
System::EventArgs^  e) {

      int sensibilidadPan_Aux=0;
      int sensibilidadTilt_Aux=0;

      if(Flag_Status_ModeSens==true){
            int SensJoy_Pan=abs(Axis_Pan)/(1000/trackBar_SensibilidadPan-
>Maximum);
            if(SensJoy_Pan!=0 && SensJoy_Pan<50){trackBar_SensibilidadPan-
>Value=SensJoy_Pan;}

            int SensJoy_Tilt=abs(Axis_Tilt)/(1000/trackBar_SensibilidadTilt-
>Maximum);
            if(SensJoy_Tilt!=0 && SensJoy_Tilt<50){trackBar_SensibilidadTilt-
>Value=SensJoy_Tilt;}

      }//if
      if(Flag_StatusPan_RIGHT==true){
            sensibilidadPan_Aux=50+(trackBar_SensibilidadPan->Value);
       }//if()
       if(Flag_StatusPan_LEFT==true){
            sensibilidadPan_Aux= 50-(trackBar_SensibilidadPan->Value);
       }//if()
       if(Flag_StatusTilt_UP==true){
            sensibilidadTilt_Aux=50+(trackBar_SensibilidadTilt->Value);
       }//if()
       if(Flag_StatusTilt_DOWN==true){
            sensibilidadTilt_Aux=50-(trackBar_SensibilidadTilt->Value);
       }//if()

       //Ejecutamos el comando Pan/tilt
       if((Flag_StatusPan_RIGHT||Flag_StatusPan_LEFT)==true){

cam_commands.Evento_PanOperation(serialPort_BUS_CAM,sensibilidadPan_Aux);
             Count_Flag_StatusPan=0;
       }//if()

       if((Flag_StatusPan_RIGHT||Flag_StatusPan_LEFT)==false){

if(Count_Flag_StatusPan==0){cam_commands.Evento_PanOperation(serialPort_BUS_CAM,
50);}
       Count_Flag_StatusPan=Count_Flag_StatusPan+1;
       }//if()

       if((Flag_StatusTilt_UP||Flag_StatusTilt_DOWN)==true){
            //retado de envio de datos por puerto
             Thread::CurrentThread->Sleep(45);

cam_commands.Evento_TiltOperation(serialPort_BUS_CAM,sensibilidadTilt_Aux);
```

```
            Count_Flag_StatusTilt=0;
        }//if()

        if((Flag_StatusTilt_UP||Flag_StatusTilt_DOWN)==false){
            if(Count_Flag_StatusTilt==0){
            //retado de envio de datos por puerto
            Thread::CurrentThread->Sleep(45);
            cam_commands.Evento_TiltOperation(serialPort_BUS_CAM,50);
        }//if()
            Count_Flag_StatusTilt=Count_Flag_StatusTilt+1;
        }//if()

}
```

<End>

6.2.4. Controles de Lentes (Zoom, Focus e Iris)

Este control será diferente para cada versión del programa, ya que utilizaremos distintos comandos para controlar las lentes de las cámaras.

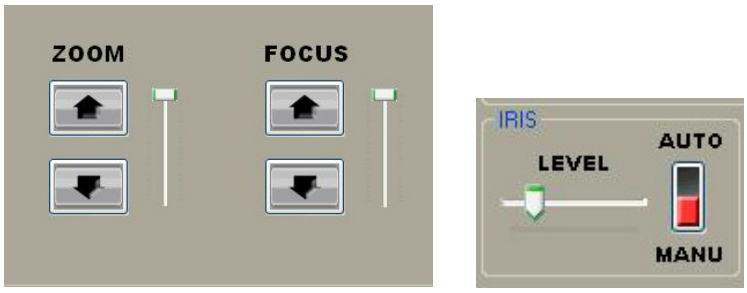


Fig. 6 y Fig. 7: "Imágenes del control del Zoom, Focus e Iris"

En la versión de programa para los modelos AW-PH300A/AW-PH360L usaremos los comandos de operación: Zoom, Focus e Iris. Estos comandos no devuelven la posición de las lentes, pero funciona en ambos.

En la versión de programa para modelos AW-PH360L, usaremos el comando 'Evento_LensCommand()' ya que nos devuelve la posición de las lentes cada vez que está siendo ejecutado, y además no produce discontinuidades en el movimiento al ejecutar de forma continua el comando.

6.2.4.1. Controles de lentes para la versión de los modelos AW-PH300A/AW-PH360L

Este control es similar al comentado en el apartado "6.2.3. Controles Pan/Til", únicamente varia el nombre de las variable, los comandos de cámara y los TrackBars.

<Código C++>

```
private: System::Void timer_LensCommands_Tick(System::Object^  sender,
System::EventArgs^  e) {

      int sensibilidadZoom_Aux=0;
      int sensibilidadFocus_Aux=0;

      if(Flag_Status_ModeSens==true){

      sensibilidadZoom_Aux=abs(Axis_Zoom)/(1000/trackBar_SensibilidadZoom-
>Maximum);
```

```
            if(sensibilidadZoom_Aux!=0 &&
sensibilidadZoom_Aux<50){trackBar_SensibilidadZoom->Value=sensibilidadZoom_Aux;}
      }//if()

       if(Flag_StatusZoom_UP==true){
            sensibilidadZoom_Aux=50+trackBar_SensibilidadZoom->Value;
      }//if()

       if(Flag_StatusZoom_DOWN==true){
                  sensibilidadZoom_Aux=50-(trackBar_SensibilidadZoom->Value);
       }//if()

       if(Flag_StatusFocus_UP==true){
            sensibilidadFocus_Aux=50+trackBar_SensibilidadFocus->Value;
       }//if()
       if(Flag_StatusFocus_DOWN==true){
            sensibilidadFocus_Aux=50-(trackBar_SensibilidadFocus->Value);
       }//if()

      if((Flag_StatusZoom_UP==true||Flag_StatusZoom_DOWN)==true){
             //Ejecutamos el comando Zoom

cam_commands.Evento_ZoomOperation(serialPort_BUS_CAM,sensibilidadZoom_Aux);
             Count_Flag_StatusZoom=0;
      }//if()

      if((Flag_StatusZoom_UP||Flag_StatusZoom_DOWN)==false){
             if(Count_Flag_StatusZoom==0){
                   //retado de envio de datos por puerto
                  Thread::CurrentThread->Sleep(45);
             cam_commands.Evento_ZoomOperation(serialPort_BUS_CAM,50);
             }//if()
             Count_Flag_StatusZoom=Count_Flag_StatusZoom+1;
       }//if()

      if((Flag_StatusFocus_UP||Flag_StatusFocus_DOWN)==true){
             //Ejecutamos el comando Focus

cam_commands.Evento_FocusOperation(serialPort_BUS_CAM,sensibilidadFocus_Aux);
            Count_Flag_StatusFocus=0;
      }//if()

      if((Flag_StatusFocus_UP||Flag_StatusFocus_DOWN)==false){
             if(Count_Flag_StatusFocus==0){
                  //retado de envio de datos por puerto
                   Thread::CurrentThread->Sleep(45);
             cam_commands.Evento_FocusOperation(serialPort_BUS_CAM,50);
             }//if()
             Count_Flag_StatusFocus=Count_Flag_StatusFocus+1;
       }//if()

}
```

<End>

6.2.4.2. Controles de lentes para la versión del modelo AW-PH360L

La estructura del código es similar que la anterior, exceptuando el comando "Evento_LensCommand()" que devuelve el valor la posición exacta de las lentes.

<Código C++>

```
private: System::Void timer_LensCommands_Tick(System::Object^  sender,
System::EventArgs^  e) {

      int sensibilidadZoom_Aux=0;
      int sensibilidadFocus_Aux=0;

      if(Flag_Status_ModeSens==true){

      sensibilidadZoom_Aux=abs(Axis_Zoom)/(1000/trackBar_SensibilidadZoom-
>Maximum);
            if(sensibilidadZoom_Aux!=0){trackBar_SensibilidadZoom-
>Value=sensibilidadZoom_Aux;}
      }//if()

       if(Flag_StatusZoom_UP==true){
            sensibilidadZoom_Aux=trackBar_SensibilidadZoom->Value;
      }//if()

       if(Flag_StatusZoom_DOWN==true){
            sensibilidadZoom_Aux=-(trackBar_SensibilidadZoom->Value);
       }//if()

       if(Flag_StatusFocus_UP==true){
            sensibilidadFocus_Aux=(trackBar_SensibilidadFocus->Value);
       }//if()
       if(Flag_StatusFocus_DOWN==true){
            sensibilidadFocus_Aux=-(trackBar_SensibilidadFocus->Value);
       }//if()


      if((Flag_StatusZoom_UP||Flag_StatusZoom_DOWN||Flag_StatusFocus_UP||Flag_St
atusFocus_DOWN)==true){
       //Ejecutamos el comando LensCommand

bufferSalidaLensCondition=cam_commands.Evento_LensCommand(serialPort_BUS_CAM,
bufferSalidaLensCondition,sensibilidadZoom_Aux,sensibilidadFocus_Aux,(LEVEL_IRIS
->Value));
      }//if()
```

<End>

6.2.5. Controles del panel Pre-set

La selección de un pre-set es bastante sencilla, utilizaremos el método "EventoCambioMem()" incluido en la cabecera "CamCommands.h".

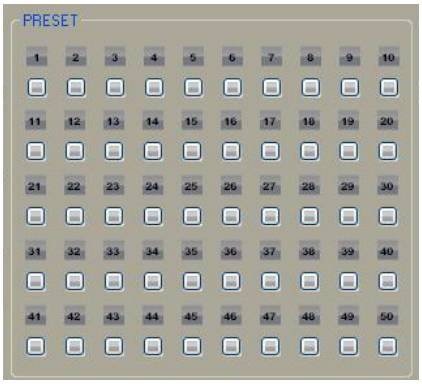


Fig. 8: "Imágenes del control de selección de Pre-set"

El procedimiento al cambiar de cámara es el siguiente:

1. Resetear las luces de selección de los Pre-set de la interfaz Gráfica y encendemos la luz del Pre-set seleccionado.
2. Guardamos en número de Pre-set seleccionado en una variable de tipo array<int>, donde estará contenido el último Pre-set seleccionado de cada cámara.

<Código C++>

```
      //Declaramos la variable BufferPresetSelected_Cam
      private: array<Int32>^ BufferPresetSelected_Cam;

private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e)
{
      //Variable BufferPresetSelected_Cam para guardar el numero de preset
selecionado para cada camara
      this->BufferPresetSelected_Cam= gcnew array<Int32>(6);

      //valor inicial BufferPresetSelected_Cam
      BufferPresetSelected_Cam[1]=1; //Valor inical Preset1 para la camara 1
      BufferPresetSelected_Cam[2]=1; //Valor inical Preset1 para la camara 2
      BufferPresetSelected_Cam[3]=1; //Valor inical Preset1 para la camara 3
      BufferPresetSelected_Cam[4]=1; //Valor inical Preset1 para la camara 4
      BufferPresetSelected_Cam[5]=1; //Valor inical Preset1 para la camara 5
}
```

<End>

3. Ejecutamos el comando para seleccionar el Pre-set.

A continuación mostramos el código del evento cambio de Pre-set para el 'Pre-set 1', será idéntico para cada uno de los 50 Pre-sets, solamente hay que cambiar el valor de entrada 'numero de Pre-Set' del comando "EventoCambioMem()" y el nombre de la clase button "button_mem1" por el del Pre-set deseado.

El código completo de todos los Pre-sets esta incluido en el archivo de código "Form1.h" del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
private: System::Void Evento_Mem1(){
      //Reset luces de los Preset
      Evento_ResetLucesPreset();

      //Encender luz de preset_1
      button_mem1->ImageIndex=1;

      //Guardamos el valor del numero de Preset seleccionado para cada camara
      BufferPresetSelected_Cam[numCamSelected]=1;//Valor numero de Preset_1

      //Seleccionamos el Preset_1
      //valor inicial Preset 1-> '0,0'
      bufferSalidaPreset=cam_commands.EventoCambioMem(bufferSalidaPreset,serialP
ort_BUS_CAM,0,0);
}
```

<End>

6.2.6. Control Pre-set Memory

Como podemos observar en el protocolo de comunicación RS-232C, el cambio de memoria y el Pre-set memory están relacionados. Únicamente varia el operador ID data 'M' a 'R' del comando **'#Maab'.**

Cuando pulsamos el botón Memory, los botones del Pre-set y del mismo botón se iluminan en rojo indicando que está en modo de grabación 'R' de la posición actual P/T y de lentes. Y a continuación pulsaremos el número de Pre-set donde queremos grabar la posición.

Clip de ejemplo: ..\Ficheros anexos\Videos\Ejemplo_funcion_MemoryPreset.avi

Este botón solo cambia el operador ID de la variable "bufferSalidaPreset" donde esta contenido el comando de cambio de Pre-set y después al presionar unos de los Pre-sets el comando ejecutado lleva el operador memory 'R' y memoriza la posición en ese pre-set.

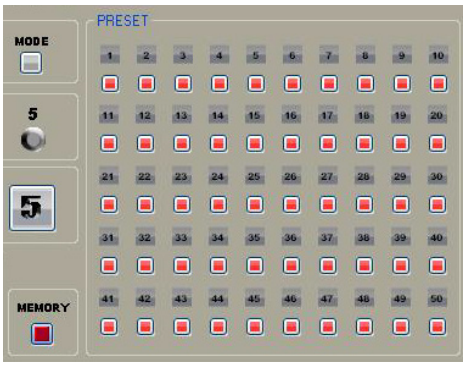


Fig. 9: "Imágenes del control de memorización de los Pre-sets"

<Código C++>

```
private: System::Void MEMORY_Click(System::Object^  sender, System::EventArgs^
e) {
      //Comando del para guardar una configuracion en un Preset -> #R00.
       switch(bufferSalidaPreset[1]){
             case 'R':

                   //Valor bit de activacion MEMORY
                   bufferSalidaPreset[1]='M';
                   //Habilitamos el boton MEMORY
                   MEMORY->ImageIndex=2;

                   //Encender luces Record de los Preset
                   Evento_LoadLucesRecordPreset();
                    break;

             case 'M':

                    //valor bit desactivacion MEMORY
                    bufferSalidaPreset[1]='R';
                   //deshabilitamos el boton MEMORY
                    MEMORY->ImageIndex=0;

                   //Apagar luces de los Preset
                    Evento_ResetLucesPreset();
             break;
       };//switch()


 }
```

<End>

6.2.7. Interruptores de condición (Condition_SW)

El establecimiento del estado de los interruptores de condición, tal y como hemos explicado en el apartado “4.2.11 Comando para establecer los controles de Condición” afecta a las siguientes funciones: EXT, ND, IRIS MODE, LAMP, SERVICE SWITCH (OP), DEF, WIPE y HEAT/FAN.

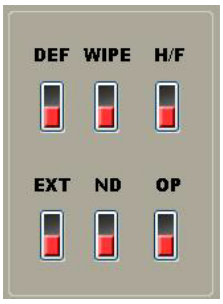


Fig. 10 y Fig. 11: “Imágenes de los interruptores de condición ”

Para establecer el estado de los interruptores necesitaremos declarar una variable de tipo array<wchar_t>, la cual contendrá el byte de estado de cada interruptor.

<Código C++>

```
      /*variables de informacion del estado del interruptor: 0:OFF y 1:ON para:
            1:EXT_SW, 2:ND_SW, 3:IRIS_SW, 4:LAMP_SW,
5:INQUIRY_of_LAMP_FILAMENT_BREAK_NONE
      6:SERVICE_SWITCH_SW, 7:DEF_SW,8:WIPE_SW y 9:HEAT/FAN_SW -->
SW_ConditionControl_Byte_b*/
private: array<wchar_t>^ SW_ConditionControl_Byte_b;

private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e)
{

       //Valor inicial SW_ConditionControl_Byte_b
       SW_ConditionControl_Byte_b[0]='0';
       SW_ConditionControl_Byte_b[1]='0';
       SW_ConditionControl_Byte_b[2]='0';
       SW_ConditionControl_Byte_b[3]='0';
       SW_ConditionControl_Byte_b[4]='0';
       SW_ConditionControl_Byte_b[5]='0';
       SW_ConditionControl_Byte_b[6]='0';
       SW_ConditionControl_Byte_b[7]='0';
       SW_ConditionControl_Byte_b[8]='0';
}
```

<End>

Al pulsar el interruptor cambiará su estado de activo/desactivo o viceversa, según su estado anterior.

Ejecutará el comando “Evento_ConditionControl” contenido del encabezado “CamCommands.h”.

Y finalmente guardará el valor de la variable ‘SW_ConditionControl_Byte_b’ para cada una de las cámaras.

En el siguiente código C++ mostramos un ejemplo de uno de los interruptores (ModeIrirs), para el resto de interruptores será igual lo único que cambiaremos es el índice de la variable ‘SW_ConditionControl_Byte_b [índice]’, el valor’ SW_Number’ de la función “Evento_ConditionControl” y el nombre de la clase del boton ‘ModeIris’ por el correspondiente interruptor.

El código completo de todos los interruptores esta incluido en el archivo de código "Form1.h" del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
private: System::Void ModeIris_Click(System::Object^  sender, System::EventArgs^
e) {
      //Comando de Modo de funcionamiento del iris -> Auto/Manual
       switch(SW_ConditionControl_Byte_b[2]){
             case '0':

                   //Valor bit de activacion IRIS_SW ->modo automatico
                   SW_ConditionControl_Byte_b[2]='1';

                   //Enviamos el evento de ConditionControl -> modo
manual/automatico

cam_commands.Evento_ConditionControl(serialPort_BUS_CAM,bufferSalidaConditionCon
trol,3,1);


                   //Habilitamos el boton ModeIris
                   ModeIris->ImageIndex=1;
                   break;
            case '1':

                    //valor bit desactivacion IRIS_SW -> modo manual
                   SW_ConditionControl_Byte_b[2]='0';

                   //Enviamos el evento de ConditionControl ->IRIS_SW -> modo
manual/automatico

cam_commands.Evento_ConditionControl(serialPort_BUS_CAM,bufferSalidaConditionCon
trol,3,0);

                   //Habilitamos el boton ModeIris
                   ModeIris->ImageIndex=0;
                   break;

       };//switch()

       //Guarda el estado actual del valor de SW_ConditionControl_Byte_b
      SaveVariablesConditionControl();
     }
```

<End>

6.2.8. Límites de posición Pan/Tilt

Esta función es bastante fácil de implementar para cada límite ejecutaremos el comando "Evento_SettingPositionLimit" del encabezado "CamCommands.h" con el valor de la variable 'Mode Limit' correspondiente para cada uno y guardaremos su valor en la variable 'int numPositionLimit'.

Fig. 11: "Imagen de los controles de Limite de posición"

<Código C++>

```
//************Botones y comandos para establecer limites de
posicion*******************************************
private: System::Void PositionLimit_Upper_Click(System::Object^  sender,
System::EventArgs^  e) {
      //introducimos el numero de boton de limite
      numPositionLimit=1;

      //Habilitamos o Deshabilitamos el límite de posicion

cam_commands.Evento_SettingPositionLimit(serialPort_BUS_CAM,1);

            }
private: System::Void PositionLimit_Under_Click(System::Object^  sender,
System::EventArgs^  e) {
      //introducimos el numero de boton de limite
      numPositionLimit=2;

      //Habilitamos o Deshabilitamos el límite de posicion

cam_commands.Evento_SettingPositionLimit(serialPort_BUS_CAM,2);
            }

private: System::Void PositionLimit_Left_Click(System::Object^  sender,
System::EventArgs^  e) {
      //introducimos el numero de boton de limite
       numPositionLimit=3;

      //Habilitamos/Deshabilitamos el límite de posicion

cam_commands.Evento_SettingPositionLimit(serialPort_BUS_CAM,3);
            }
private: System::Void PositionLimit_Right_Click(System::Object^  sender,
System::EventArgs^  e) {
      //introducimos el numero de boton de limite
      numPositionLimit=4;

      //Habilitamos/Deshabilitamos el límite de posicion

cam_commands.Evento_SettingPositionLimit(serialPort_BUS_CAM,4);
                        }
```

<End>

Esta variable 'numPositionLimit' la utilizaremos para saber el estado activo/inactivo de cada límite que nos responde el cabeza P/T por medio del comando "Resultados del comando para Establecer/ Resetear el límite de posición" del protocolo de comunicación RS-232C, recibido por el Puerto Serie.

El siguiente código lo incluiremos en el evento "DataReceived" de la clase 'SerialPort', el cual se ejecutará cada vez que se reciban datos por el puerto serie.

El código consiste en:

1. Lee el dato recibido por el puerto serie.
2. Comprueba que es el comando de resultado del estado del límite.
3. Deshabilita/Habilita el botón de límite seleccionado.
4. Guarda el valor de estado del límite seleccionado para cada cámara, en una variable de tipo array<wchar_t>:

<Código C++>

```
//Variable BufferStatusLimit
private: array<wchar_t>^ BufferStatusLimit;

//variable de BufferStatusLimit
 BufferStatusLimit=gcnew array<wchar_t>(4);

 //Valor inicial de BufferStatusLimit
 BufferStatusLimit[0]='0';
 BufferStatusLimit[1]='0';
 BufferStatusLimit[2]='0';
 BufferStatusLimit[3]='0';
```

<End>

En el siguiente fragmento de código, mostramos el código completo del evento DataReceived:

<Código C++>

```
private: System::Void serialPort_BUS_CAM_DataReceived(System::Object^  sender,
System::IO::Ports::SerialDataReceivedEventArgs^  e) {

      //borramos la variable bufferEntrada_RX
       bufferEntrada_RX->Clear(bufferEntrada_RX,0,3);

      //Metodo utilizado para recibir los comandos de resultados del BUS_MEM_RX
y contestar el cambio de camara
       serialPort_BUS_CAM->Read(bufferEntrada_RX,0,3);

      if(bufferEntrada_RX[1]==0x31){
            switch(numPositionLimit){

                  case 1:
                        //Activamos la luz de PositionLimit_Upper
                        PositionLimit_Upper->ImageIndex=1;

                        //Habilitamos el flag BufferStatusLimit
                        BufferStatusLimit[0]='1';
                        break;

                  case 2:
                        //Activamos la luz de PositionLimit_Under
                        PositionLimit_Under->ImageIndex=1;

                        //Habilitamos el flag BufferStatusLimit
                        BufferStatusLimit[1]='1';

                        break;

                  case 3:
                        //Activamos la luz de PositionLimit_Left
                        PositionLimit_Left->ImageIndex=1;

                        //Habilitamos el flag BufferStatusLimit
                        BufferStatusLimit[2]='1';

                        break;


                  case 4:
                        //Activamos la luz de PositionLimit_Right
                        PositionLimit_Right->ImageIndex=1;
                        //Habilitamos el flag BufferStatusLimit
                        BufferStatusLimit[3]='1';
                        break;
```

```
                default:

                        break;

        };//Switch()

    }else{
        switch(numPositionLimit){

                case 1:
                        //Desactivamos la luz de PositionLimit_Upper
                        PositionLimit_Upper->ImageIndex=0;

                        //Deshabilitamos el flag BufferStatusLimit
                        BufferStatusLimit[0]='0';
                        break;

                case 2:
                        //Desactivamos la luz de PositionLimit_Under
                        PositionLimit_Under->ImageIndex=0;

                        //Deshabilitamos el flag BufferStatusLimit
                        BufferStatusLimit[1]='0';

                        break;


                case 3:
                        //Desactivamos la luz de PositionLimit_Left
                        PositionLimit_Left->ImageIndex=0;

                        //Deshabilitamos el flag BufferStatusLimit
                        BufferStatusLimit[2]='0';

                        break;


                case 4:
                        //Desactivamos la luz de PositionLimit_Right
                        PositionLimit_Right->ImageIndex=0;

                        //Deshabilitamos el flag BufferStatusLimit
                        BufferStatusLimit[3]='0';

                        break;

                default:

                        break;
        };//Switch()
    }//if()

}
```

<End>

6.2.9. Panel de ajustes de posición Pan/Tilt y Lentes

Este panel de ajustes solo está incluido en la versión para la cabeza P/T "AW-PH360L", ya que los comandos utilizados para establecer la posición no están habilitados para el cabezal "AW-PH300A".

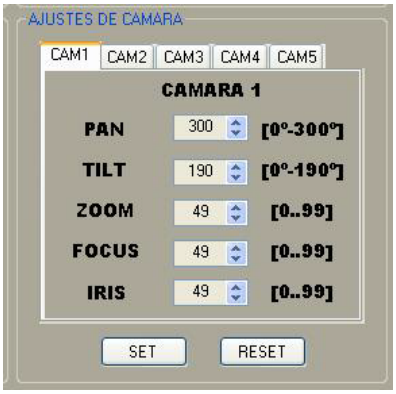


Fig. 12: "Imágenes del panel de ajustes de posición de operadores de cámara"

Hemos diseñado una interfaz grafica que está formada por los siguientes componentes o herramientas de visual Studio:

1. Un ' TabControl' esta herramienta controla y muestra al usuario una colección relacionada de fichas que pueden contener controles y componentes. Nosotros introduciremos 5 fichas una para cada cámara.
2. Dentro de cada ficha añadiremos cinco componentes 'NumericUpDown' uno para cada valor de posición Pan/Tilt y de lentes. A través de esta herramienta introduciremos los valores que deseemos establecer. También nos mostrará el valor de posición actual de P/T y lentes cada vez que ejecutemos un comando de posición (Pan, Tilt, Zoom, Focus e Iris).
3. Añadiremos 2 botones 'Button'. Uno para establecer la posición introducida en los componentes 'NumericUpDown' (SET). Y otro para resetear la posición y los valores Pan/Tilt y Lentes (RESET). Ambos se ejecutarán únicamente para la cámara seleccionada en el 'TabControl'.

Cada vez que ejecutemos uno de estos botones 'SET/RESET' cargaremos los valores en los monitores (Pan, Tilt, Zoom, Focus e Iris) y en el 'Trackbar LEVEL_IRIS'. Y ejecutaremos los comandos: Evento_SetPosition() y Evento_ LensCondition().

En el siguiente código incluimos un ejemplo del botón 'SET' para la cámara 1, para el reto de cámaras será igual solo variaremos el nombre de los componentes 'NumericUpDown' por los correspondientes para cada cámara. También va incluido el código del Botón de 'RESET'.

El código completo para todas las cámaras esta incluido en el archivo de código "Form1.h" del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
//************Boton y comando para  establecer las posiciones
(Pan,Tilt,Zoom,Focus,IRIS)**********************************
private: System::Void SET_Click(System::Object^  sender, System::EventArgs^  e)
{
      switch(tabControlAjustes->SelectedIndex){
            case 0:
                  //cerramos puertos
                  this->serialPort_BUS_CAM->Close();

                  //Cambiamos el nombre de los puertos
                  this->serialPort_BUS_CAM->PortName = NombreCom_Cam1;

                  //Abrimos puertos
                  this->serialPort_BUS_CAM->Open();

                  //ejecutamos el comando setposition
 bufferSalidaSettingPosition=cam_commands.Evento_SetPosition(serialPort_BUS_CAM,
(int)(numericPanCam1->Value),(int)(numericTiltCam1->Value));

                 //cargamos la posicion pan/tilt en los monitores Pan y Tilt
                 LoadMonitor_PanTilt((int)(numericPanCam1-
>Value),(int)(numericTiltCam1->Value));

                 // Retardo de 200ms
                  Thread::CurrentThread->Join(200);

                  //Ejecutamos el comando Lenscondition

bufferSalidaLensCondition=cam_commands.Evento_LensCondition(serialPort_BUS_CAM,
(unsigned int)(numericZoomCam1->Value),(unsigned int)(numericFocusCam1-
>Value),(unsigned int)(numericIrisCam1->Value));

                  //cargamos la posicion de las lentes en los monitores Zoom,
Focus y Iris
                 LoadMonitor_LensBar((unsigned int)(numericZoomCam1-
>Value),(unsigned int)(numericFocusCam1->Value),(unsigned int)(numericIrisCam1-
>Value));

                  //Cargamos el valor en el trackbar LEVEL_IRIS
                 LEVEL_IRIS->Value=(unsigned int)(numericIrisCam1->Value)*41;

                 break;


                 };//switch()


            }
//************Boton y comando para resetear las posiciones
(Pan,Tilt,Zoom,Focus,IRIS) y los límites de posicion*************
private: System::Void RESET_Click(System::Object^  sender, System::EventArgs^
e) {
      //Valor inicial bufferSalidaSettingPosition
      bufferSalidaSettingPosition[0]=0x23;
      bufferSalidaSettingPosition[1]=0x55;
      bufferSalidaSettingPosition[2]=0x37;//Bytes aaaa -> PanPosition
      bufferSalidaSettingPosition[3]=0x46;
      bufferSalidaSettingPosition[4]=0x42;
      bufferSalidaSettingPosition[5]=0x43;
      bufferSalidaSettingPosition[6]=0x37;//Bytes bbbb -> TiltPosition
      bufferSalidaSettingPosition[7]=0x46;
      bufferSalidaSettingPosition[8]=0x41;
      bufferSalidaSettingPosition[9]=0x38;
      bufferSalidaSettingPosition[11]=0x0D;//delimiter (Current-CR)
```

```
        for(int i=2;i<10;i++){ //Byte d -> checkSum 8 byte data

              bufferSalidaSettingPosition[10]=bufferSalidaSettingPosition[10]+buff
        erSalidaSettingPosition[i];
        }//for()

        //Valor inicial bufferSalidaLensCondition
        bufferSalidaLensCondition[0]=0x23;
        bufferSalidaLensCondition[1]=0x41;
        bufferSalidaLensCondition[2]=0x37;//Bytes aaa -> ZoonPosition
        bufferSalidaLensCondition[3]=0x44;
        bufferSalidaLensCondition[4]=0x39;
        bufferSalidaLensCondition[5]=0x37;//Bytes bbb -> FocusPosition
        bufferSalidaLensCondition[6]=0x44;
        bufferSalidaLensCondition[7]=0x39;
        bufferSalidaLensCondition[8]=0x37;//Bytes ccc -> IrisPosition
        bufferSalidaLensCondition[9]=0x44;
        bufferSalidaLensCondition[10]=0x39;
        bufferSalidaLensCondition[12]=0x0D;//delimiter (Current-CR)
        for(int i=2;i<=10;i++){ //Byte d -> checkSum 8 byte data

              bufferSalidaLensCondition[11]=bufferSalidaLensCondition[11]+bufferSa
        lidaLensCondition[i];
        }//for()

        //Ejecutamos el comando Pan/tilt

bufferSalidaSettingPosition=cam_commands.Evento_PanTiltCommand(serialPort_BUS_CA
M, bufferSalidaSettingPosition,0,0);


        //Establecemos la posicion actual de los monitores PAN/TILT
        Evento_Establecer_Monitor_PanTiltPosition();

        //retado de envio de datos por puerto
         Thread::CurrentThread->Sleep(100);

         //Ejecutamos el comando LensCommand

bufferSalidaLensCondition=cam_commands.Evento_LensCommand(serialPort_BUS_CAM,
bufferSalidaLensCondition,0,0,2047);

        //Cargamos el valor en el trackbar LEVEL_IRIS
        LEVEL_IRIS->Value=2047;

        //Establecemos la posicion actual de los monitores LensCondition
        Evento_Establecer_Monitor_LensCondition();
}
```

<End>

6.2.10. Save/Load

Diseñaremos distintos métodos para cargar y guardar el último estado de:

- La configuración Serial Port de cada cámara, es decir el nombre del puerto asignado para cada cámara.
  Void LoadConfiguracionSerialPort().

- El estado de los interruptores de condición para cada una de las cámaras.
  Void LoadVariablesConditionControl().
  Void SaveVariablesConditionControl()

- El último Pre-set seleccionado en cada una de las cámaras.
  Void LoadUltimaSeleccionPreset()

- El valor de todos los componentes 'NumericUpDown' de Pan/Tilt y Lentes para la cámara seleccionada.
  Void LoadNumericPosition(unsigned int PanPosition_Grados, unsigned int TiltPosition_Grados,int CamSelected);
  Void LoadNumericLensCondition(unsigned int ZoomPosition,unsigned int FocusPosition,unsigned int IrisPosition,int CamSelected);

- El valor de todos los componentes TrackBars de sensibilidad Pan/Tilt y Lentes para todas las cámaras.
  Void LoadTrackBarPosition();
  Void SaveTrackBarPosition();

- Los valores de posición de Pan/Tilt y Lentes.
  Void LoadPanTiltPosition();
  Void SavePanTiltPosition();
  Void LoadLensCondition();
  Void SaveLensCondition();

- El estado de los botones Limit para cada cámara.
  LoadVariablesStatusLimit();
  Void SaveVariablesStatusLimit();

Todos estos valores serán guardados en un archivo .TXT para cada método.

Generalmente ejecutaremos los métodos Load/Save cada vez que abramos/cerremos la aplicación y cuando cambiamos de cámara.

También utilizaremos los métodos Save para guardar los valores cada vez que ejecutamos una función de programa como: el Pan/Tilt, el Zoom, un interruptor, etc.

El código de estos métodos SAVE/LOAD estan incluidos en el archivo de código "Form1.h" del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

6.2.11. Monitores Pan/Tilt y Lentes

Este control únicamente estará incluido en la versión de programa para el modelo AW-PH360L.

Hemos diseñado unos monitores gráficos que muestran la posición aproximada de la cabeza P/T y las lentes.

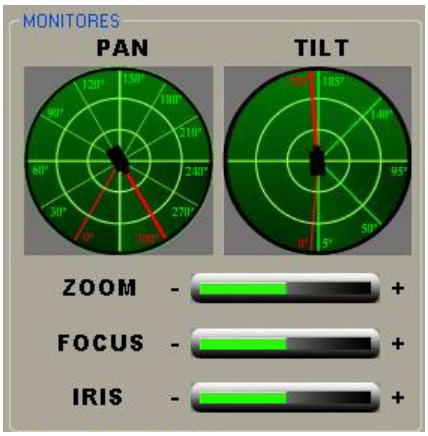


Fig. 13: "Imagen de los monitores de posición de Pan/tilt y lentes "

Las siguientes funciones están basadas en un componente "pictureBox" donde se cargarán las imágenes según la posición introducida.

Hemos creado tantas imágenes como valores de posición se posee cada componente. Por ejemplo, el Pan tiene un margen de movimiento de 0º a 300º, por lo tanto deberemos crear 301 imágenes una por ángulo. Así para todas las funciones.

Las imágenes las he realizado por medio del Adobe Flash y las he exportado a una secuencia de imágenes. Esta secuencia la hemos numerado con el mismo margen de valores de posición.

Las imágenes de los monitores están contenidas en las carpetas:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\Monitor Pan
..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\MonitorTilt
..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\Monitor Bar

<Código C++>

```
private: System::Void LoadMonitor_PanTilt(unsigned int Num_Image_Pan,unsigned
int Num_Image_Tilt){

      if(Num_Image_Pan<=300){
            String^ fileToDisplay="../Monitor
Pan/animacion_monitor_pan"+(Num_Image_Pan+1)+".PNG";
            pictureBox_Pan->Image=Image::FromFile(fileToDisplay);
       }//if

       if(Num_Image_Tilt<=195){
            String^
fileToDisplay="../MonitorTilt/animacion_monitor_tilt"+(Num_Image_Tilt+1)+".PNG";
            pictureBox_Tilt->Image=Image::FromFile(fileToDisplay);
```

```
        }//if
 }
private: System::Void LoadMonitor_LensBar(unsigned int Num_Image_Zoom,unsigned
int Num_Image_Focus,unsigned int Num_Image_Iris){

        if(Num_Image_Zoom<=100){
              String^ fileToDisplay="../Monitor
Bar/Animacion_MonitorBar"+(Num_Image_Zoom+1)+".PNG";
              pictureBox_Zoom->Image=Image::FromFile(fileToDisplay);
        }//if

        if(Num_Image_Focus<=100){
              String^ fileToDisplay="../Monitor
Bar/Animacion_MonitorBar"+(Num_Image_Focus+1)+".PNG";
              pictureBox_Focus->Image=Image::FromFile(fileToDisplay);

        }//if

        if(Num_Image_Iris<=100){
              String^ fileToDisplay="../Monitor
Bar/Animacion_MonitorBar"+(Num_Image_Iris+1)+".PNG";
              pictureBox_Iris->Image=Image::FromFile(fileToDisplay);

        }//if
 }
```

<End>

Ambos métodos los ejecutamos desde los eventos:

- Evento_Establecer_Monitor_PanTiltPosition();
- Evento_Establecer_Monitor_LensCondition();

Estos métodos introducen y calculan el valor actual de posición en grados para cada uno en los monitores P/T.

<Código C++>

```
private: System::Void Evento_Establecer_Monitor_PanTiltPosition(){
        //Declaramos variables
         unsigned int PanPosition=0;
         unsigned int TiltPosition=0;

        //Convertimos los valores_PanPosition de Hexadecimal a decimal
        PanPosition=HexToDecimal(bufferSalidaSettingPosition,2,5);

        //Convertimos los valores_TiltPosition de Hexadecimal a decimal
        TiltPosition=HexToDecimal(bufferSalidaSettingPosition,6,9);

        //Convertimos los valores decimales en grados Pan/tilt
       PanPosition=PanPosition/218;
       TiltPosition=TiltPosition/344;

       //Cargamos la imagen del angulo correspondiente
       LoadMonitor_PanTilt(PanPosition,TiltPosition);


       LoadNumericPosition(PanPosition,TiltPosition,numCamSelected);
 }

private: System::Void Evento_Establecer_Monitor_LensCondition(){
        //Declaramos variables
          int ZoomPosition=0;
          int FocusPosition=0;
          int IrisPosition=0;

        //Convertimos los valores_ZoomPosition de Hexadecimal a decimal
        ZoomPosition=HexToDecimal(bufferSalidaLensCondition,2,4);
```

```
        //Convertimos los valores_FocusPosition de Hexadecimal a decimal
        FocusPosition=HexToDecimal(bufferSalidaLensCondition,5,7);


        //Convertimos los valores_IrisPosition de Hexadecimal a decimal
        IrisPosition=HexToDecimal(bufferSalidaLensCondition,8,10);

        //Normalizamos los valores LensCondition
        //Y cargamos los valores LensCondition en numericLensCondition
      LoadNumericLensCondition(ZoomPosition/41,
      FocusPosition/41,IrisPosition/41,numCamSelected);

      //Mostramos la posicion en el monitor de barra -> Monitor_LensBar
      LoadMonitor_LensBar(ZoomPosition/41, FocusPosition/41,IrisPosition/41);
 }
```

<End>

Ambos métodos los ejecutaremos por medio de un 'Timer' ("timer_Update_Monitors") con un intervalo de tiempo de 100ms.

Como hemos explicado en el apartado "6.2.3 Controles Pan/Tilt", al no poder utilizar la función 'Evento_PanTiltCommand()' repetidas veces ya que se producen movimientos torpes e discontinuos. Hemos tenido que calcular de una manera aproximada la nueva posición al ejecutar un movimiento Pan/Tilt.

Este cálculo de la nueva posición lo podemos realiza ya que cuando encendemos las cámaras o cambiamos de cámara se introduce el valor de la última posición tanto en la cabeza P/T como en el programa. En cambio esto no vale para los modelos AW-300A porque no contienen el comando 'Evento_PanTiltCommand()'.

En cambio con la función 'Evento_LensCommand()' no pasa, por lo tanto la posición de las lentes será la que nos proporcione esa función y será la posición exacta de las lentes.

Considerando que para la sensibilidad de Pan y Tilt son iguales a 25, la cámara se mueve un grado más o menos. Y el factor equivalente de desplazamiento para Pan es K=14 y para Tilt k=22.

Este factor lo multiplicaremos por la sensibilidad de cada uno y nos proporcionará el desplazamiento, que sumaremos a la posición anterior. Esta operación se realizará solamente si se está pulsando un de los controles Pan/tilt. Si no se pulsa ningún control mostrará la posición P/T actual.

En el evento 'timer_Update_Monitors_Tick()' además de ejecutar los eventos para establecer la posición, también calcularemos la nueva posición P/T.

<Código C++>

```
private: System::Void timer_Update_Monitors_Tick(System::Object^  sender,
System::EventArgs^  e) {

        //Convertimos los valores_PanPosition de Hexadecimal a decimal
        int PanPosition=HexToDecimal(bufferSalidaSettingPosition,2,5);

        //Convertimos los valores_TiltPosition de Hexadecimal a decimal
        int TiltPosition=HexToDecimal(bufferSalidaSettingPosition,6,9);


        //Calculamos  el equivalente en grados del movimiento segun su
sensibilidad para el PAN->1°==sensibilidadPan=25 -> Desplazamiento=218 aprox.
      if((Flag_StatusPan_RIGHT==true)){
            PanPosition=PanPosition+(14*(trackBar_SensibilidadPan->Value));
```

```
        }//if()

        if((Flag_StatusPan_LEFT==true) ){
              PanPosition=PanPosition-(14*(trackBar_SensibilidadPan->Value));
        }//if()

      //Calculamos  el equivalente en grados del movimiento segun su
sensibilidad para el Tilt->1°==sensibilidadTilt=25 -> Desplazamiento=345 aprox.
        if(Flag_StatusTilt_UP==true){
              TiltPosition=TiltPosition+(22*(trackBar_SensibilidadTilt->Value));
        }//if()

        if(Flag_StatusTilt_DOWN==true){
              TiltPosition=TiltPosition-(22*(trackBar_SensibilidadTilt->Value));
        }//if()

        if ((PanPosition>0)&& (PanPosition<65534)){
      //Pasamos de decimal a hexadecimal y guardamos en
bufferSalidaSettingPosition

bufferSalidaSettingPosition=DecimalToHex(bufferSalidaSettingPosition,
PanPosition,2,5);
      }//if

        if ((TiltPosition>0)&& (TiltPosition<65534)){
        //Pasamos de decimal a hexadecimal y guardamos en
bufferSalidaSettingPosition

bufferSalidaSettingPosition=DecimalToHex(bufferSalidaSettingPosition,
TiltPosition,6,9);

      }//if
       //Establecemos la posicion actual de los monitores PAN/TILT
      Evento_Establecer_Monitor_PanTiltPosition();

      //Establecemos la posicion actual de los monitores LensCondition
      Evento_Establecer_Monitor_LensCondition();

}
```

<End>

6.2.12. Menú

El menú está formado con el componente “StripMenuItem”, donde incluiremos las opciones de la aplicación según su funcionalidad (Opciones, Ayuda, etc)

En la pestaña de ayuda hemos incluido el manual de la aplicación “Manual.pdf”y un dialogBox ‘AcercaDe’ que contiene información sobre: el autor, contacto, licencia…

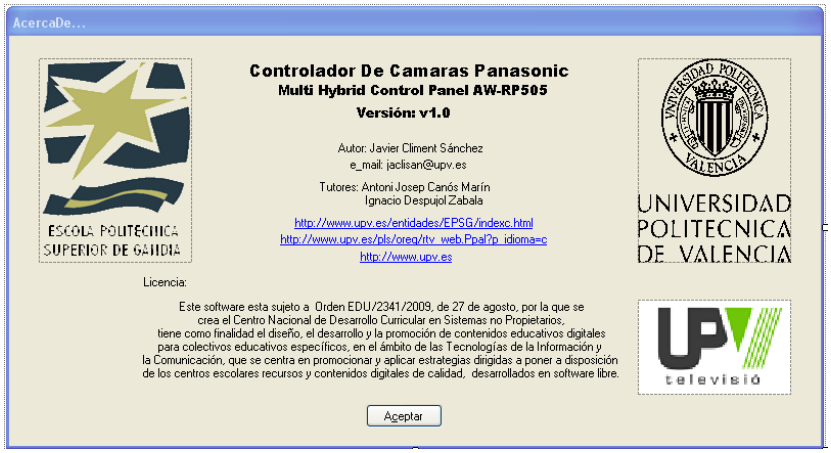


Fig. 14 y Fig. 15: “Imágenes de la barra de menú y de la caja de dialogo AcercaDe…”

Y en la pestaña de Opciones: un DialogBox destinado a la configuración de los puertos serie ‘Configuración Serial Port’ y el evento ‘Salir’ para cerrar la aplicación.

Fig. 16 y Fig. 17: “Imágenes de la barra de menú y de la caja de dialogo Configuración de Puertos serie“

A continuación mostramos el código de los eventos menú para integrar los DialogBox en el formulario principal:

<Código C++>

```
//******************Eventos del StripMenuItem*****************************

private: System::Void acercaDeToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {

       Form ^ f = (gcnew Acerca::AcercaDe());
      f->ShowDialog();

 }

private: System::Void salirToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       Close();
 }
```

```
private: System::Void manualToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {

       try{
             Process::Start("Manual.pdf");
       }
       catch(Win32Exception^ ex){
             MessageBox::Show("No se encuentra el archivo: 'Manual.pdf' \n"+
                              "Asegúrese de que el achivo se encuentra en la
misma ubicación"+
                              " que la aplicación","Aviso:"+ ex->ToString(),
MessageBoxButtons::OK,MessageBoxIcon::Warning);

       }

 }
private: System::Void
configuraciónPuertosComToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       Form^ f2 = (gcnew controlCAM_c::ConfiguracionPuertosSerie());
       f2->ShowDialog();

       if(f2->DialogResult==Windows::Forms::DialogResult::OK){

             LoadConfiguracionSerialPort();

       }//if()

 }
```

<End>

El código de estos DialogBox estan incluidos en los archivo de código “AcercaDe.h” y “ConfiguracionPuertosSerie.h”del proyecto Visual C++:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\AcercaDe.h

Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\ConfiguracionPuertosSerie.h

6.2.13. Teclado

La integración del teclado en la aplicación la realizaremos a través de los eventos ‘KeyDown’ y ’KeyUp’ del formulario principal ‘Form1’.

‘KeyDown’ se ejecutará cuando presionemos alguna tecla, mientras que ‘KeyUp’ se ejecutará cuando soltemos la tecla.

Tenemos que tener en cuenta que cuando presionemos una tecla para ejecutar un función debemos inhabilitar el timer ‘timer_UpdateStick’ del joystick, porque si no se produce una interferencia. Y cuando soltemos la tecla volveremos a encenderlo.

A continuación mostramos un fragmento del código necesario para declarar una tecla. Si se desea visualizar el código completo, abrir el archivo “Form1.h”:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
private: System::Void Form1_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e) {

      //En este metodo introduciremos los controles del teclado para cada evento

       // Apagamos el timer para la actualizacion del estado de joystick
       timer_UpdateStick->Enabled = false;

      //Controles de teclado para eventos Pan/Tilt
      if(System::Console::CapsLock==false){

            if(e->KeyCode==Keys::W){

                  Evento_UpDown();
            }//if
}

private: System::Void Form1_KeyUp(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e) {

       if(System::Console::CapsLock==false){

                        if(e->KeyCode==Keys::W){

                              Evento_UpUp();
                        }//if
}
```

<End>

6.2.14. Ratón

Todos los componentes de la interfaz gráfica como: los botones y TrackBars poseen eventos para la integración del ratón:

- 'Click'-> Tiene lugar cuando se hace click en el componente.
- 'MouseDown'-> Tiene lugar cuando se presiona el componente con el botón del ratón.
- ' MouseUp'-> Tiene lugar cuando se suelta el botón del mouse.

El siguiente código muestra un fragmento de la implementación del ratón en nuestra aplicación, si se desea podemos ver el código completo en:

..\Ficheros anexos\Software y cabeceras\controlCAM_c++_ToHead_UltimaVersion_AW-PH360L\controlCAM_c++\Form1.h

<Código C++>

```
private: System::Void button_cam1_Click(System::Object^  sender,
System::EventArgs^  e) {
      //ejecutamos el evento cambio a la camara 1
       Evento_Cam1();
            }
private: System::Void UP_MouseDown(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {
     // Apagamos el timer para la actualizacion del estado de joystick
     timer_UpdateStick->Enabled = false;

       Evento_UpDown();
}
```

```
private: System::Void UP_MouseUp(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {

        Evento_UpUp();

        // Encendemos el timer para la actualizacion del estado de joystick
        timer_UpdateStick->Enabled = true;

 }
```

<End>

6.2.15. Joystick

Para el desarrollo del dispositivo 'Joystick' utilizaremos el Framework 'SlimDX' de código abierto gratuito, que permite a los desarrolladores crear fácilmente aplicaciones DirectX utilizando .NET como C #, VB.NET y C++.

Añadiremos a nuestro proyecto visual C++ una librería de clases, donde le agregaremos un nuevo elemento 'Control de Usuario'. Este proyecto al compilarlo nos proporcionará una .Dll (librería dinámica) que utilizaremos en nuestra aplicación como si fuera un componente más.

Fig. 18: "Imagen del botón de activación del joystick"

Al nuevo elemento 'Control de Usuario' tenemos que incluirle la referencia al Framework SlimDX:

- Vamos al menú (Proyecto→Referencias…)
- Pulsamos "Agregar Referencia"
- Buscamos el archivo 'SlimDX.dll'en la carpeta: ..\SlimDX\Bin\net20\x86
- Incluimos el namespace→"`using namespace SlimDX::DirectInput;`"

Ahora ya podemos declarar las variables y clases del Joystick:

<Código C++>

```
private: Joystick^ joystick;
private: JoystickState^ state;
private: int numPOVs;
private: int SliderCount;
```

<End>

Dentro del control de usuario desarrollaremos el código de Joystick está estructurado de la siguiente forma:

1. Declararemos variables de propiedad para poder introducir y obtener los valores de información del estado de los controles del dispositivo: ejes, botones, POV…

<Código C++>

```
private: int axisX;
   /// <summary>
/// Primer eje del joystick
   /// <summary>
public:     property int AxisX {
              int get() {return axisX;}
              void set(int axis){axis=axisX;}
          }
```

```
  private: int axisY;
    /// <summary>
  /// Segundo eje del joystick
    /// <summary>
  public:     property int AxisY {
                int get() {return axisY;}
                void set(int axis){axis=axisY;}
            }
  private: int axisZ;
    /// <summary>
  /// Tercer eje del joystick
    /// <summary>
  public:     property int AxisZ {
                int get() {return axisZ;}
                void set(int axis){axis=axisZ;}
            }
  private: int axisXRot;
    /// <summary>
  /// Cuarto eje del joystick
    /// <summary>
  public:     property int AxisXRot {
                int get() {return axisXRot;}
                void set(int axis){axis=axisXRot;}
              }
  private: int axisYRot;
    /// <summary>
  /// Quinto eje del joystick
    /// <summary>
  public:     property int AxisYRot {
                int get() {return axisYRot;}
                void set(int axis){axis=axisYRot;}
              }
  private: int axisZRot;
    /// <summary>
  /// sexto eje del joystick
    /// <summary>
  public:     property int AxisZRot {
                int get() {return axisZRot;}
                void set(int axis){axis=axisZRot;}
            }

  private: int POV;
  /// <summary>
  /// Array de eje POV del Joystick.
/// <summary>
  public:     property int Pov {
                int get() {return POV;}
  /*            void set(array<int>^ pov_aux){//pov->CopyTo(POV,0);
                }
  */          }

  private: array<int>^ slider;
  /// <summary>
  /// Array de eje slider del Joystick.
/// <summary>
  public:     property array<int>^ Slider {
                array<int>^ get() {return slider;}
                void set(array<int>^ SLIDER){//SLIDER->CopyTo(slider,0);
                }
              }
```

```
private:bool Button1;
  /// <summary>
/// Variable de boton habilitado del Joystick.
  /// <summary>
public:     property bool BUTTON1 {
              bool get() {return Button1;}
              void set(bool BUTTONS){
                        Button1=BUTTONS;
              }
         }
private:bool Button2;
  /// <summary>
/// Variable de boton habilitado del Joystick.
  /// <summary>
public:     property bool BUTTON2 {
              bool get() {return Button2;}
              void set(bool BUTTONS){
                        Button2=BUTTONS;
              }
         }
private:bool Button3;
  /// <summary>
/// Variable de boton habilitado del Joystick.
  /// <summary>
public:     property bool BUTTON3 {
              bool get() {return Button3;}
              void set(bool BUTTONS){
                        Button3=BUTTONS;
              }
         }
private:bool Button4;
  /// <summary>
/// Variable de boton habilitado del Joystick.
  /// <summary>
public:     property bool BUTTON4 {
              bool get() {return Button4;}
              void set(bool BUTTONS){
                        Button4=BUTTONS;
              }
         }
private:bool Button5;
  /// <summary>
/// Variable de boton habilitado del Joystick.
  /// <summary>
public:     property bool BUTTON5 {
              bool get() {return Button5;}
              void set(bool BUTTONS){
                        Button5=BUTTONS;
              }
                  }
```

<End>

2. Crearemos un método para la declarar el dispositivo. El cual primero busca los dispositivos conectados, crea una instancia del mismo y finalmente activa el dispositivo.

<Código C++>

```
public: void CreateDevice()
        {
      /// <summary>
                  // Crea un dispositivo
      /// <summary>

      // Inicializamos la clase DirectInput
            DirectInput^ dinput = gcnew DirectInput();



      // Busca los dispositivos conectados
      for each (DeviceInstance^ device in dinput-
>GetDevices(DeviceClass::GameController, DeviceEnumerationFlags::AttachedOnly))
            {
            // Crea el dispositivo
              try
               {
                joystick = gcnew Joystick(dinput, device->InstanceGuid);
                              joystick->SetCooperativeLevel(this,
CooperativeLevel::Exclusive | CooperativeLevel::Foreground);
                break;
                }
                catch (DirectInputException^ err)
                {
                }
            }

                  if (!joystick)
            {
                        MessageBox::Show("There are no joysticks attached to the
system.");
                return;
            }

            for each (DeviceObjectInstance deviceObject in joystick-
>GetObjects())
            {
                  if (((int)deviceObject.ObjectType &
(int)ObjectDeviceType::Axis)!= 0)
                              joystick-
>GetObjectPropertiesById((int)deviceObject.ObjectType)->SetRange(-1000, 1000);

                UpdateControl(deviceObject);
            }

            // Activa el dispositivo
            joystick->Acquire();

            //Activamos el Timer de lectura de estado del Joytick
            timer_JOY->Start();
                        }
```

<End>

3. Creamos otro método para la lectura de los datos recibidos por el dispositivo.

<Código C++>

```
public:void ReadImmediateData()
        {
        /// <summary>
                //Lee el estado de los botones del Joystick
        /// <summary>
           if (joystick->Acquire().IsFailure)
                return;

            if (joystick->Poll().IsFailure)
                return;

            state = joystick->GetCurrentState();
                  if (SlimDX::Result::Last.IsFailure)
                return;

            UpdateUI();
       }
```

<End>

4. Un método para liberar el dispositivo

<Código C++>

```
public: void ReleaseDevice()
        {
        /// <summary>
                //Libera o desactiva el joystick
        /// <summary>
            timer_JOY->Stop();

            if (!joystick)
            {
                joystick->Unacquire();
                //joystick->Dispose();
            }
        }
```

<End>

5. Crearemos dos métodos más: uno para leer el estado actual de de cada control del dispositivo y otro que comprueba el estado de los controles.

<Código C++>

```
public: void UpdateControl(DeviceObjectInstance d)
        {
        /// <summary>
      /*Comprueba el estado Enable/Disable de los ejes y botones
                */
        /// <summary>
      if (ObjectGuid::XAxis == d.ObjectTypeGuid)
            {
            Status_Xaxis=true;
            }
      if (ObjectGuid::YAxis == d.ObjectTypeGuid)
            {
            Status_Yaxis=true;
            }
      if (ObjectGuid::ZAxis == d.ObjectTypeGuid)
            {
            Status_Zaxis=true;
            }
```

```
        if (ObjectGuid::RotationalXAxis == d.ObjectTypeGuid)
              {
              Status_RotationalXaxis=true;
              }
        if (ObjectGuid::RotationalYAxis == d.ObjectTypeGuid)
              {
              Status_RotationalYaxis=true;
              }
        if (ObjectGuid::RotationalZAxis == d.ObjectTypeGuid)
              {
              Status_RotationalZaxis=true;
              }
        if (ObjectGuid::Slider == d.ObjectTypeGuid)
             {
              Status_Slider=true;
              }
        if (ObjectGuid::PovController == d.ObjectTypeGuid)
              {
              Status_PovController=true;
              }
        if (ObjectGuid::Button == d.ObjectTypeGuid)
              {
              Status_Button=true;
              }

}
```

<End>

6. Añadiremos un componente Timer con un intervalo de 100ms, que ejecutará el método de lectura de datos del dispositivo.

<Código C++>

```
private: System::Void timer_JOY_Tick(System::Object^  sender, System::EventArgs^
e) {
                    ReadImmediateData();
        }
```

<End>

7. Crearemos un botón para activar/desactivar el joystick a través del comando 'Enable' de la clase "timer_JOY" .

<Código C++>

```
private: System::Void createDeviceButton_Click(System::Object^  sender,
System::EventArgs^  e) {
         /// <summary>
         /// <summary>
        if (timer_JOY->Enabled==false){
              createDeviceButton->Text = "ON";
              createDeviceButton->ImageIndex=1;
              timer_JOY->Enabled=true;
        }else{
              createDeviceButton->Text = "OFF";
              createDeviceButton->ImageIndex=0;
              timer_JOY->Enabled=false;
                    }
}
```

<End>

8. Y finalmente introduciremos en el evento Load del control de usuario, el código necesario para que se ejecute el joystick y declararemos la variables iníciales.

<Código C++>

```
private: System::Void ControlJoy_Load(System::Object^  sender,
System::EventArgs^  e) {
             state = gcnew JoystickState();
             POV=-1;
             if (!joystick){
                  CreateDevice();
                  createDeviceButton->Text = "ON";
                  createDeviceButton->ImageIndex=1;
             }else{
                  ReleaseDevice();
                  UpdateUI();
                  createDeviceButton->Text = "OFF";
                  createDeviceButton->ImageIndex=0;
             }
 }
```

<End>

De la misma forma que el control de Joystick que hemos creado lee los datos en cada momento por medio de un Timer, declararemos un Timer en nuestra aplicación que lea la variable de propiedad que nos proporciona el control de joystick. Y además ejecute los eventos de cámara (movimiento P/T, Lentes y cambio de cámara), según el elemento del joystick que está siendo pulsado.

<Código C++>

```
private: System::Void timer_UpdateStick_Tick(System::Object^  sender,
System::EventArgs^  e) {

                   //Lee el estado de los elementos de joystick
                  Axis_Pan=controlJoy1->AxisX;
                  Axis_Tilt=controlJoy1->AxisY;
                  Axis_Zoom=controlJoy1->AxisZRot;

                  Axis_Focus = -1;
                  Axis_Focus = controlJoy1->Pov;

                  bool buttonsState1 = controlJoy1->BUTTON1;
                  bool buttonsState2 = controlJoy1->BUTTON2;
                  bool buttonsState3 = controlJoy1->BUTTON3;
                  bool buttonsState4 = controlJoy1->BUTTON4;
                  bool buttonsState5 = controlJoy1->BUTTON5;

                  //Ejecuta los comandos según el elemento activo del joystick
                  if(Axis_Pan>300){
                         Evento_RightDown();
                  }//if()
                  if(Axis_Pan<300) {
                         Evento_RightUp();
                  }//if()

                  if(Axis_Pan<-300){
                         Evento_LeftDown();
                  }//if()
                  if(Axis_Pan>-300){
                         Evento_LeftUp();
                  }//if()
```

```
	if(Axis_Tilt>300){
		Evento_DownDown();
	}//if()
	if(Axis_Tilt<300){
		Evento_DownUp();
	}//if()
	if(Axis_Tilt<-300){
		Evento_UpDown();
	}//if()
	if(Axis_Tilt>-300){
		Evento_UpUp();
	}//if()

	if(Axis_Zoom>300){
		Evento_ZoomWide_Down();
	}//if()

	if(Axis_Zoom<300){
		Evento_ZoomWide_Up();
	}//if()

	if(Axis_Zoom<-300){
		Evento_ZoomTele_Down();
	}//if()

	if(Axis_Zoom>-300){
		Evento_ZoomTele_Up();
	}//if()

	switch(Axis_Focus){
		case 0:
		 Evento_FocusFar_Down();
			break;

		case 18000:
		 Evento_FocusNear_Down();
			break;

		case -1:
		 Evento_FocusFar_Up();
		 Evento_FocusNear_Up();
			break;

		default:
			break;
	};
	if (buttonsState1==true){
		this->Evento_Cam1();
	}//if()
	if (buttonsState2==true){
		this->Evento_Cam2();
	}//if()
	if (buttonsState3==true){
		this->Evento_Cam3();
	}//if()
	if (buttonsState4==true){
		this->Evento_Cam4();
	}//if()
	if (buttonsState5==true){
		this->Evento_Cam5();
	}//if()
}
```

<End>

7. Integración del Sistema de cámaras con el software Video Toaster.

La idea para integrar el sistema de cámaras con el software (Video Toaster) es poder automatiza los ajustes de Pre-set preajustados con los escenarios virtuales de la herramienta LiveSet, a través de una tabla de configuración que por ejemplo contenga la información de los comandos de operación de cámaras (de número de Pre-set y número de cámara seleccionada) y para cada uno de estos operadores un escenario y un ajuste de vista del escenario:

Para el programa… ej. "Caminos en el aire"				
Cámara	**Pre-set**	**escenario**	**Vista del escenario**	**Entrada del Bus Preview Seleccionado**
1	4	Caminos en el aire 1	cerca	1
3	1	Caminos en el aire 2	derecha	3
2	8	Caminos en el aire 1	frontal	2

El funcionamiento sería el siguiente:

- Cuando se presionará algún botón de Bus Preview del Mezclador de la VT, guardamos el valor del Preview que contiene la cámara que se desea seleccionar.

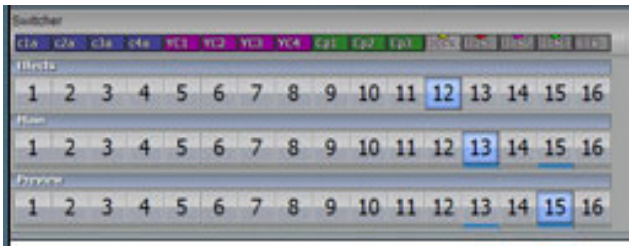


Fig. 1: "Imagen de los Buses del Switcher de Video Toaster"

- Después seleccionamos un escenario (Scene) con su ajuste de vista del escenario (Shot), guardaremos los valores de ambos y se ejecuta un callback (devolución de llamada) que compararía los valores seleccionados en la VT con dicha tabla.

Fig. 2: "Imagen del Live-Set de Video Toaster"

- Según el resultado de las comparaciones se seleccionará la cámara y el pre-set establecidos en la tabla, para el escenario (Scene), su ajuste de vista (Shot) y la cámara del Bus PREVIEW seleccionada en la VT.

Esto reduciría considerablemente las operaciones realizadas por el operador técnico. Además daría más fiabilidad a la hora de establecer los ajustes de las cámaras para un escenario.

NewTek Video Toaster es un programa de edición lineal que combina hardware y software para la edición y producción de definición estándar y alta definición de vídeo en NTSC ,PAL .

Fig. 3: "Imagen software Video Toaster"

Se compone de varias herramientas para la conmutación de vídeo , croma , LiveSet (Escenarios Virtuales) , animación y manipulación de imágenes .

El código del software de la Video Toaster está compuesto por dos lenguajes de programación.

Uno es el Toaster Script es parecido al JavaScript, pero es poco flexible ya que posee pocos métodos y clases. Además está desarrollado para controlar únicamente los componentes, periféricos (Teclado, ratón, controlador MIDI) y su aspecto definidos en la interfaz gráfica y declarar plugins.

Estos plugins están programados con el SDK VT 5.2, cuya plataforma está desarrollada en C++.

Los principales inconvenientes de este SDK_VT5.2 son:

- No posee documentación alguna y los ejemplos son escasos y apenas tienen comentarios.
- Está compuesto por todas las versiones de SDK_VT (v.1.0 hasta v.5.0) y parcheadas una encima de las otras.
- Únicamente hay una persona de contacto que es bastante reacia a dar información sobre el SDK.

Hay noticias desde Newtek de que la nueva versión de Video Toaster posee ya un SDK_VT con documentación, pero para los modelos antiguos ya no se desarrolla ni se aporta más soporte.

Todos estos inconvenientes imposibilitan, de momento, el poder desarrollar programas con estás versiones de SDK sin documentación.

Lo máximo que se ha podido conseguir es:

- Corregir los errores del SDK_VT5.2 para poder empezar a programar. Mucho de los errores estaban producidos por direcciones erróneas de los encabezados y por la falta de estos SDK's: boost_1_40_0, IPL y wxWidgets-2.8.10.

- La creación de un plug-in vacio con su correspondiente ventana.

- La carga de un nuevo plug-in en el software VT, creado por nosotros.

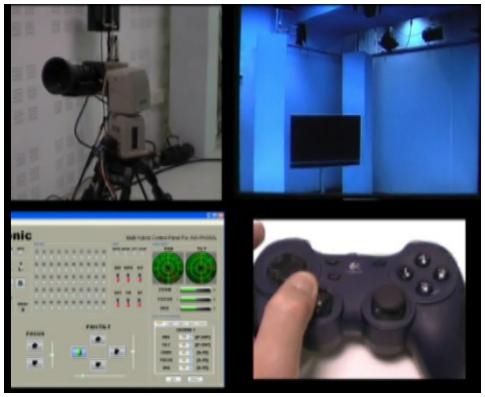
- Y la definición de su acceso a través de la barra de menú.

El código del plugin VT está contenido en la carpeta:
Ficheros anexos\Software y cabeceras\Software para VideoToaster\flagsVT_FINAL

8. Videos

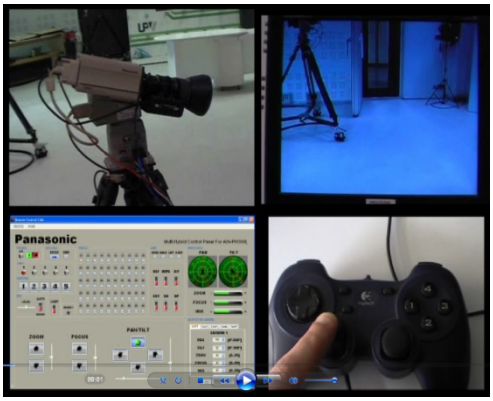
En este apartado vamos a mostrar unos videos de ejemplos, en los cuales podremos observar el funcionamiento de algunas operaciones de las cámaras conectadas a un PC y controladas a través de un dispositivo joystick.

Video ejemplo del Movimiento Pan:



Ficheros anexos\Videos\VideoEjemplo_MovimientoPan.m2v

Video ejemplo del Movimiento Tilt:



Ficheros anexos\Videos\VideoEjemplo Movimiento Tilt.m2v

Video ejemplo del Movimiento Pan y Tilt simultáneo:

Ficheros anexos\Videos\VideoEjemplo Movimiento Pan y Tilt simultaneos.m2v

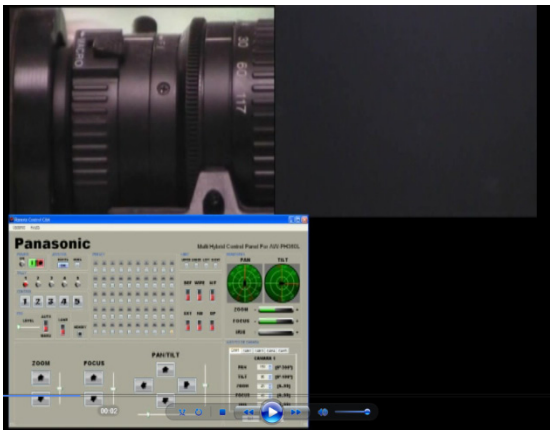
Video ejemplo del Movimiento Zoom:

Ficheros anexos\Videos\VideoEjemplo_Zoom.m2v

Video ejemplo del Movimiento Focus:

Ficheros anexos\Videos\Video ejemplo Movimiento Focus.m2v

Video ejemplo del Movimiento Iris:



Ficheros anexos\Videos\VideoEjemplo Movimieno Iris.m2v

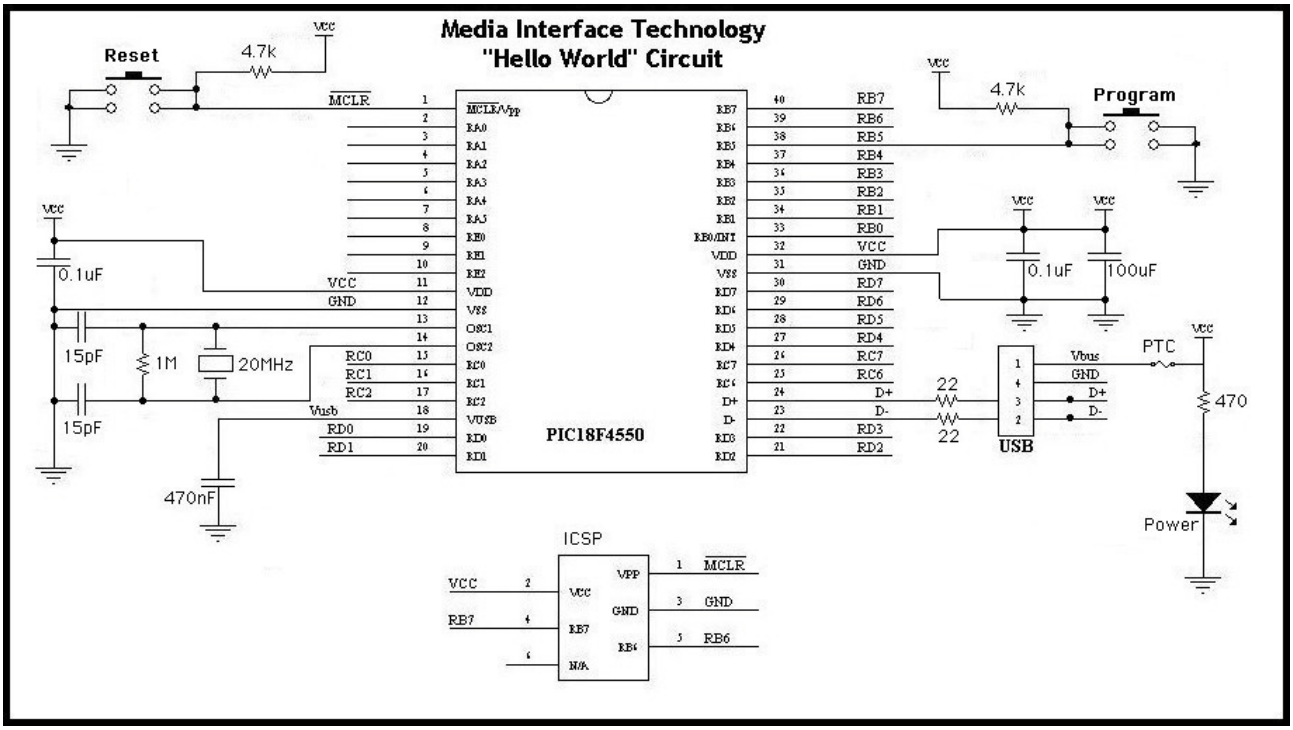
9. Futuras Aplicaciones

En este apartado vamos a explicar algunas de las futuras aplicaciones que se pueden aplicar en el sistema de cámaras.

Nuestro software se puede usar como segundo panel de cámaras, para el control simultaneo de dos cámaras. Además por medio del software podemos realizar transiciones de movimiento P/T y de lentes constantes, cosa que con el panel no se podía hacer por su abrupta variación de velocidad.

Este software puede utilizarse para la conexión de nuevos dispositivos de control personalizados, como podría ser el diseño de un servo-control de cámaras. Su hardware podría ser el circuito de un joystick y conectarlo a nuestro software por medio de la dll del Joystick que hemos diseñado en nuestro programa.

En la siguiente imagen se muestra un esquema de un circuito para conectar dispositivos analógicos a un USB:



También podemos diseñar un sistema informatizado de escenarios virtuales 3D, que este basado en un software de escenarios virtuales 3D que interactué con los movimientos de posición, zoom y enfoque de cámara, es decir, por detección de movimiento por ejes de cámara.

Esto último solo podríamos realizarlo con el modelo AW-PH360L de cabezales P/T, ya que como hemos explicado anteriormente los modelos anteriores como el AW-PH300A no poseen los comandos necesarios para establecer una posición determinada por valores de ángulo P/T o posición de las lentes.

Fig. 1 y Fig. 2: "Imágenes de un plató virtual y el escenario virtual aplicado"

10. Conclusiones

A lo largo de este proyecto hemos realizado el desarrollo de dos software y el diseño de dos interfaces, uno por cada protocolo de comunicación que hay en el sistema de cámaras, para poder integrar los escenarios Virtuales del software Video Toaster con el sistema de cámaras robóticas Panasonic.

Para llegar a esto primero debíamos realizar un software que permitirá conectar las cámaras al PC y después desarrollar otro software que nos permitiera integrar los métodos de control de las cámaras con los controles del Bus de Preview del modulo switcher de la Video Toster y sus escenarios virtuales. De forma que cuando ejecutáramos uno de los controles de la Video Toaster se ejecutará un programa que compararía los controles seleccionados en la Videotoaster con una tabla de configuración predefinida, que contuviera para que escenario y canal de Preview ejecutaríamos los comandos de selección de cámara y el preajuste Pre-set correspondiente.

En un principio intentamos desarrollar el software y el interface para el protocolo de comunicación RS-422 que conecta el Pc con el Hub de cámaras, porque evitaba tener que realizar un nuevo cableado en el estudio. Pero este sistema de control de las cámaras falló, no podíamos controlar la selección del cambio de todas las cámaras, únicamente podíamos controlar la cámara 1. El error viene producido, como hemos explicado en al final del apartado "6.1. Software para RS-422", por consecuencia de un error de sincronización del envío del comando de Tally con el resto de las secuencias de comandos enviadas a través de los buses (BUS _TX_P/T, BUS_RX_P/T), ya que los puertos series a velocidades menores de 100 ms no son tan precisos al enviar los comandos. También por la "Señal de control de cámara" enviada por el panel a través del conector "Camera Control IN/OUT" que valida el comando de TALLY por medio de una secuencia sincronizada de 47 ms con el comando de Tally formada por el valor '00'. Este problema no sucede con la cámara 1 porque el comando de Tally para esa cámara no se envía y en este caso funciona perfectamente.

Luego al ver que no funcionaba este sistema, comenzamos a desarrollar el software para el protocolo de comunicación RS232C, que va directamente del PC a las cabezas de cámara. Este protocolo tiene el único inconveniente de que debemos realizar un cableado en el plató. Pero posee muchas ventajas respecto al protocolo de comunicación anterior, como:

- El desarrollo de este software está basado en el protocolo de comunicación RS-232. Es mucho más sencillo de programar, porque sus comandos actúan como eventos y no se envía contantemente una secuencia de comandos sincronizada, tal y como sucede en el Protocolo de comunicación RS-422.

- 50 Pre-sets, que con el Protocolo RS-422 y el panel únicamente poseemos 10 Pre-Sets.

- La posibilidad de establecer una posición exacta Pan/Tilt y de Lentes en grados (sólo para cabezales AW-PH360L). Esto nos permitirá poder saber la posición de las cámaras en todo momento y desarrollar funciones para cargar/guardar la posición actual de cada cámara además de monitorizar gráficamente la posición de la cabeza P/T y de las lentes.

Como solo podemos utilizar esta última función para los cabezales P/T "AW-PH360L" desarrollaremos dos versiones del software, uno para ambos modelos (AW-PH300A/AW-PH360L) y otra versión solo para el modelo "AW-PH360L". La versión de Software para el modelo "AW-PH300A" también es válida para el modelo "AW_PH360L", pero no a la inversa.

Ambas versiones son muy parecidas, ya que poseen muchas funciones en común del protocolo de comunicación. Únicamente difieren en:

- Los controles de posición del Pan/Tilt y Lentes.
- La monitorización y establecimiento de la posición de las cámaras.
- La Carga y el Save de las posiciones del cabezal P/T y las lentes en un archivo txt.

Y al final logramos diseñar un Software con dos versiones, una para cada modelo de cabezal P/T (AW-PH300A y AW-PH360L que son los cabezales que tenemos en el plató de la televisión de la UPV), que integraban el sistema de cámaras robóticas con el PC.

Este software será utilizado en la televisión de la UPV, para tener el panel y el PC como dos controladores del sistema de cámaras, lo que nos proporcionará el poder manejar dos cámaras a la vez en la producción de programas. Además una clara ventaja sobre el Panel, es que el software tiene la propiedad de poder realizar transiciones de movimiento Pan/Tilt y lentes con una velocidad constante establecida por el usuario, cosa que con el panel no sucede ya que las palancas de control tienen una sensibilidad variable y no están preciso como para realizar planos en movimiento a una velocidad constante.

En el desarrollo de un plugin que integrará las cámaras con el programa de la Video Toaster nos encontramos con los siguientes problemas en su SDK_VT 5.2:

- No hay documentación alguna y los ejemplos son escasos y apenas tienen comentarios.

- Está compuesto por todas las versiones de SDK_VT (v.1.0 hasta v.5.0) y parcheadas una encima de las otras, esto provoca que antes de poder empezar a compilar se produzcan innumerables errores. Pero esto último conseguimos solucionarlo y los hemos introducido en un SDK VT corregido, que está en el siguiente directorio: ..\Ficheros anexos\Software y cabeceras\SDK VT5\SDK VT5.2 (Corregido sin errores y Completo)\VT52_SDK2.rar

- Únicamente hay una persona de contacto que es bastante reacia a dar información sobre el SDK. Ya que hay noticias desde Newtek , de que la nueva generación de Video Toaster posee ya un SDK_VT con documentación, pero para los modelos antiguos ya no se desarrolla ni se aporta más soporte.

Y finalmente hay que incidir que la versión del software desarrollado para los modelos 'AW-360L' de cabeza P/T, tiene comandos que establecen la posición del movimiento Pan/Tilt y lentes y por lo tanto podemos saber en todo momento en que posición está la cámara (en grados) y la posición de las lentes (del Zoom, Focus e Iris). Esto permitirá desarrollar aplicaciones para diseñar un sistema informatizado de escenarios virtuales 3D, que este basado en un software de escenarios virtuales 3D que interactué con los movimientos de posición, zoom y enfoque de cámara, es decir, por detección de movimiento por ejes de cámara y la ubicación de las mismas en el estudio. Esto quedaría mucho más realista que lo se pretendía con la integración del software de la Video toaster, ya que el escenario se movería y enfocaría según actuarán las cámaras.

11. Bibliografía

Protocolos de comunicación:
..\Ficheros anexos\Documentación\Protocolos de comunicación\Convertible _protocol_ V2_33_20090518_Open.pdf
..\Ficheros anexos\Documentación\Protocolos de comunicación\Convertible_ Controllers_ command.pdf
..\Ficheros anexos\Documentación\Protocolos de comunicación\Protocolo RS232C CAMARA.pdf
..\Ficheros anexos\Documentación\Protocolos de comunicación\Protocolo RS232C PAN-TILT.pdf
..\Ficheros anexos\Documentación\Protocolos de comunicación\Protocolo RS422 PAN-TILT (panel).pdf

Manuales del sistema de cámaras Panasonic:
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\AW-E655-manual-638.pdf
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\AW-ph360L.pdf
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\AWPH360N.PDF
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\AW-RP555L.pdf
Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\control _camaras_ AW-RP505.pdf
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\m_aw-ph360le.pdf
..\Documentación\Manuales del sistema de camaras Panasonic\manual de instrucciones aw-ph500.PDF
Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\manual de usuario panel camaras aw-bp555n.PDF
..\Ficheros anexos\Documentación\Manuales del sistema de camaras Panasonic\MultiportHubAW_ HB505.pdf

Tabla ASCII:
..\Ficheros anexos\Documentación\ASCII Table - ASCII character codes and html, octal, hex and decimal chart conversion.htm

Paginas de cómo crear un dispositivo analógico:
http://www.create.ucsb.edu/~dano/CUI/
http://denki.world3.net/arcade.html

El estándar RS-232
Ficheros anexos\Documentación\EL ESTÁNDAR RS-232 y V24.docx

Pagina de desarrolladores MSDN:
http://msdn.microsoft.com/es-es/default

Manuales ToasterScript
..\Ficheros anexos\Documentación\ToasterScript\ToasterScriptDocs2.doc
..\Ficheros anexos\Documentación\ToasterScript\ToasterScriptDocs3.odt (versión OpenOffice)
..\Ficheros anexos\Documentación\ToasterScript\ToasterScriptDocs3[1].pdf

Manual Conversor i485
..\Ficheros anexos\Documentación\RS-232 to RS422 RS485 Interface Converter.pdf

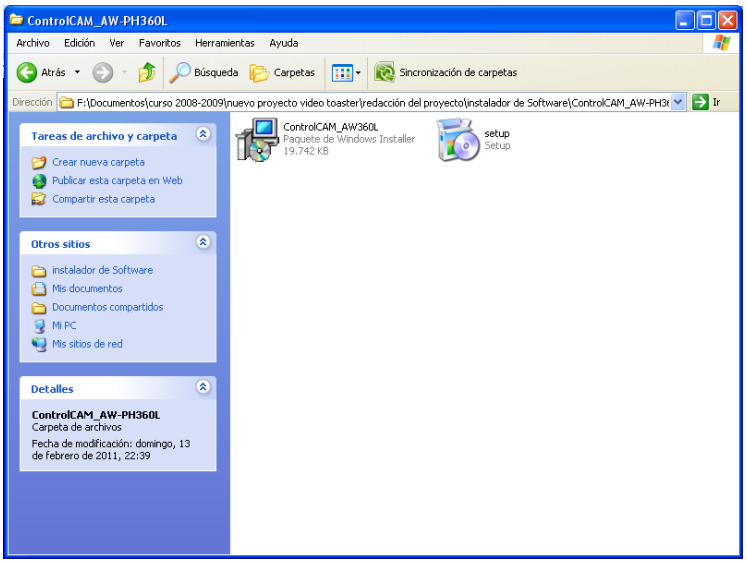
Manuales tarjeta remota de relés "K8056"
..\Ficheros anexos\controlador de relés K8056\K8056 relay card.pdf
..\Ficheros anexos\controlador de relés K8056\manual de montaje.PDF
..\Ficheros anexos\controlador de relés K8056\manual de montaje_2.PDF

Página web y foro Newtek
http://www.newtek.com/
http://www.newtek.com/forums/

12. Anexo I: Manual Software

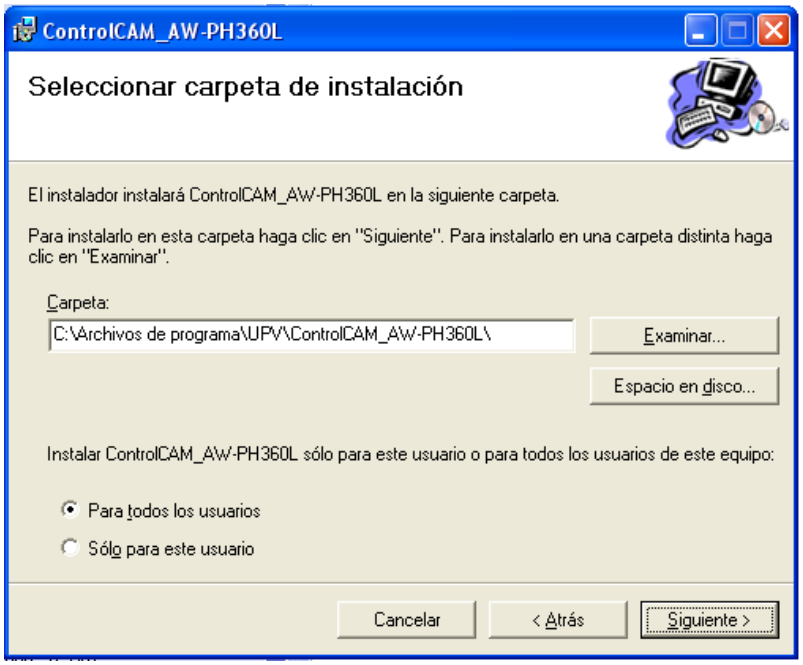
12.1. Instalación

1. Ejecutamos el archivo “setup.msi”

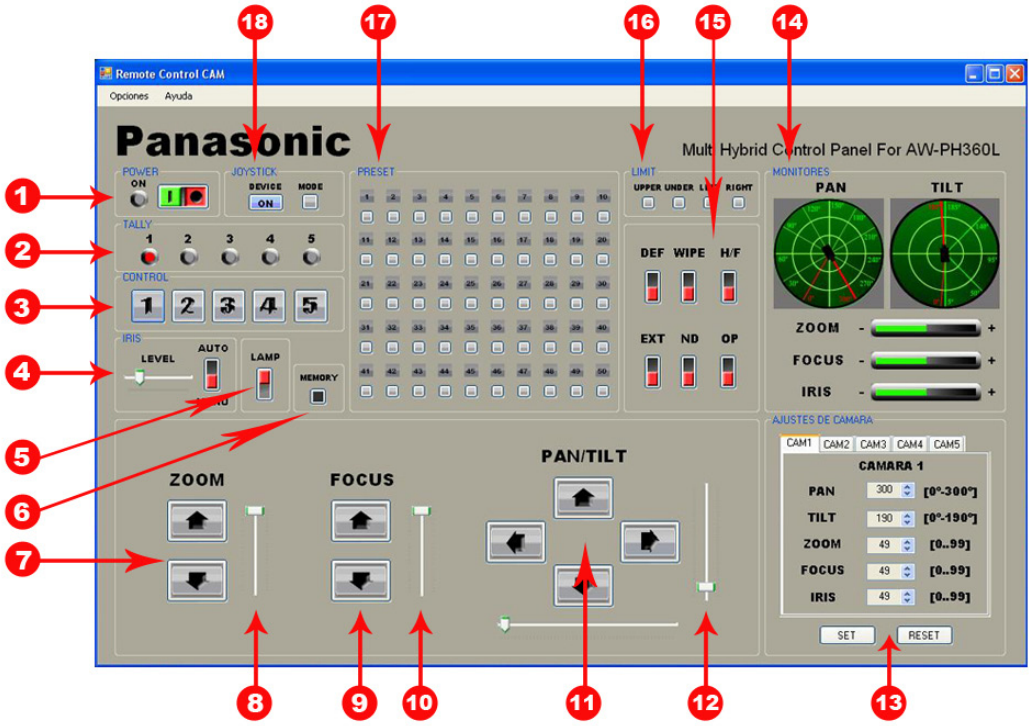


Instalación: Ficheros anexos\instalador de Software

2. Y finalmente, introducimos el directorio de instalación:



12.2. Controles de Operación y sus funciones



1. **Interruptor de encender/Apagar (POWER)**
   Enciende y apaga todas las cámaras habilitadas.

2. **Indicador de Tally**
   Muestra la cámara que esta seleccionada por la aplicación.

3. **Selector de cámara/cabeza P/T**
   Selecciona la cámara deseada con su cabeza P/T, entre las que hay conectadas al panel de control.

4. **Control y selector del Iris**
   - **Control del Iris** -> Cuando el selector del Iris está en la posición MANU), el iris puede controlarse manualmente.
   - **Selector de Iris** -> Se emplea para seleccionar el modo AUTO o MANU. Cuando el selector este en posición AUTO, el Iris se controlará automáticamente con la cantidad de luz que se introduce en el objetivo. Si esta en MANU podemos controlarlo manualmente

5. **Interruptor de lámpara (LAMP)**
   Enciende y apaga la lámpara halógena conectada al receptáculo de CA del adaptador CA (AW-PS300) de la cabeza P/T.

6. **Interruptor de la memoria de Pre-set(MEMORY)**
   Las posiciones de la cabeza P/T, Zoom/Enfoque/Iris y el equilibrio de blancos (ATW, canal A y canal B) pueden preajustarse hasta en 50 botones de la memoria por cada cabeza P/T.

Para preajustarlo en estos botones, primero seleccione una cámara con el selector de cámara (3) y establecemos la posición y el balance de blancos deseado. Presione el botón MEMORY (los botones de Pre-set se enciende en rojo), después presionamos el botón de Pre-set donde deseamos guardar el preajuste.

7. Palanca de Zoom (ZOOM,TELE/WIDE)

Se emplea para controlar la operación del zoom de la cabeza P/T seleccionada. Cuando movemos la palanca hacia arriba el zoom va hacia Teleobjetivo TELE, mientras que hacia abajo va hacia angular WIDE.

8. Sensibilidad Zoom

Establece la sensibilidad fija deseada para la velocidad de movimiento de zoom que deseamos, a no ser que para el joystick este habilitada el interruptor MODE_SENS.

9. Palanca de Enfoque (FOCUS, FAR/NEAR)

Se emplea para ajustar el enfoque del objetivo. Cuando movemos la palanca hacia arriba el enfoque va hacia FAR (lejos), en cambio cuando va hacia abajo va hacia NEAR (cercano).

10. Sensibilidad Enfoque

Establece la sensibilidad fija deseada para la velocidad de movimiento de enfoque que deseamos, a no ser que para el joystick este habilitada el interruptor MODE_SENS.

11. Palanca de Panoramización/Inclinación (PAN/TILT, UP/DOWN/LEFT/RIGHT)

Se emplea para controlar la operación panorámica e inclinación de la cabeza.

12. Sensibilidad Pan/Tilt

Los dos TracksBar establecen la sensibilidad fija deseada para la velocidad de movimiento de P/T que deseamos, a no ser que para el joystick este habilitada el interruptor MODE_SENS.

13. Ajustes de cámara

Establece los ajustes de posición P/T y lentes (Zoom, Focus e Iris) para unos valores dados previamente introducidos en las casillas de valores de posición.

Si queremos establecer estos ajustes de posición pulsamos SET. Y si queremos resetear la posición de P/T y lentes pulsamos RESET.

14. Monitores

Estos monitores muestran la posición actual de P/T y las lentes (Zoom, Enfoque e Iris).

15. Interruptores de Condición

- **Deshelador (DEF) ->** Activará/Desactivará el deshelador.
- **Limpiador (WIPE) ->** Activará/Desactivará el limpiador.
- **Calefactor/Ventilador (H/F) ->** Activará/Desactivará el Calefactor o el ventilador.
- **Extensor del objetivo (EXT) ->** Activará/Desactivará el extensor del objetivo.
- **Filtro (ND) ->** Activará/Desactivará el filtro ND.
- **Opcional (OP)** -> Controla el terminal del interruptor opcional del adaptador de CA (AW-PS300) de la cabeza P/T para cortocircuitarlo o abrirlo.

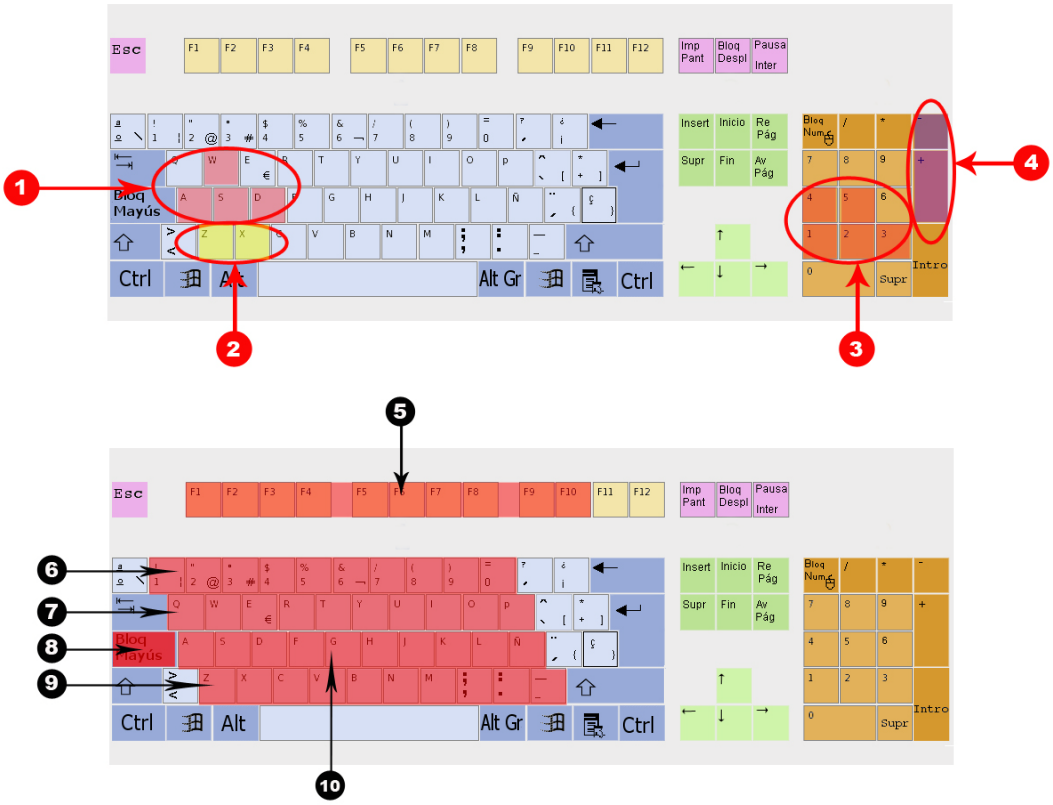
16. Límites (LIMIT, UPPER/UNDER/RIGHT/LEFT)

Estos botones se emplean para establecer un margen de movimiento P/T, es decir establece límites de movimiento en la posición actual de la cabeza P/T.

17. **Selector de la posición de preajuste (PRESET)**
    Establece los preajustes guardados en el botón de pre-set pulsado.

18. **Joystick**
    - Interruptor ModeSens -> estables si el movimiento de las palancas van a tener movilidad fija o variable (por medio del joystick).
    - Interruptor Device -> Activa o desactiva el joystick.

12.3. Controles del Teclado



1. **Controles P/T**
    - Tecla ‘W’-> Up.
    - Tecla ‘S’ -> Down.
    - Tecla ‘A’ -> Left.
    - Tecla ‘D’ -> Right.

2. **Controles Zoom**
    - Tecla ‘Z’ -> TELE.
    - Tecla ‘X’ -> WIDE.

3. **Controles para el selector de cámara**
    - Tecla ‘1’ -> Cámara 1.
    - Tecla ‘2’ -> Cámara 2.
    - Tecla ‘3’ -> Cámara 3.
    - Tecla ‘4’ -> Cámara 4.
    - Tecla ‘5’ -> Cámara 5.

4. **Controles Focus**
    - Tecla ‘+’ -> FAR.
    - Tecla ‘-’ -> NEAR.

5. **Botones del selector de Pre-set del 1 al 10 -> BloqMayús (8) + Tecla (Pre-set).**
6. **Botones del selector de Pre-set del 11 al 20-> BloqMayús (8) + Tecla (Pre-set).**
7. **Botones del selector de Pre-set del 21 al 30-> BloqMayús (8) + Tecla (Pre-set).**
8. **BloqMayús -> Moderador de teclado para las teclas del selector de Preset.**
9. **Botones del selector de Pre-set del 41 al 50-> BloqMayús (8) + Tecla (Pre-set).**
10. **Botones del selector de Pre-set del 31 al 40-> BloqMayús (8) + Tecla (Pre-set).**

12.4. Controles del Joystick

1. **Selecciona la cámara 5**
2. **Palanca de enfoque**
3. **Palanca P/T**
4. **Palanca Zoom**
5. **Selecciona la cámara 2**
6. **Selecciona la cámara 3**
7. **Selecciona la cámara 4**
8. **Selecciona la cámara 1**

12.5. Solución de problemas

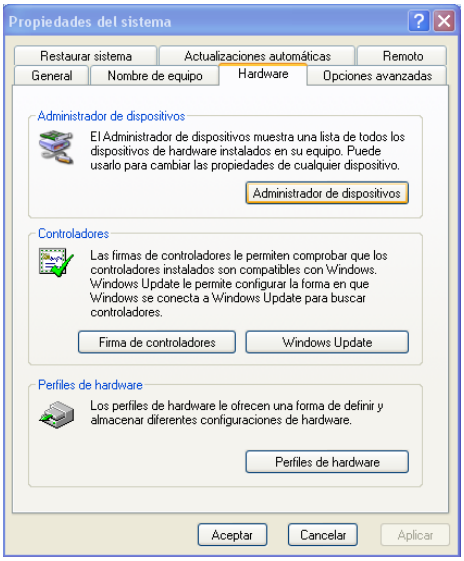
12.5.1. Mensaje de error: “El puerto ‘COM 1’ no existe”

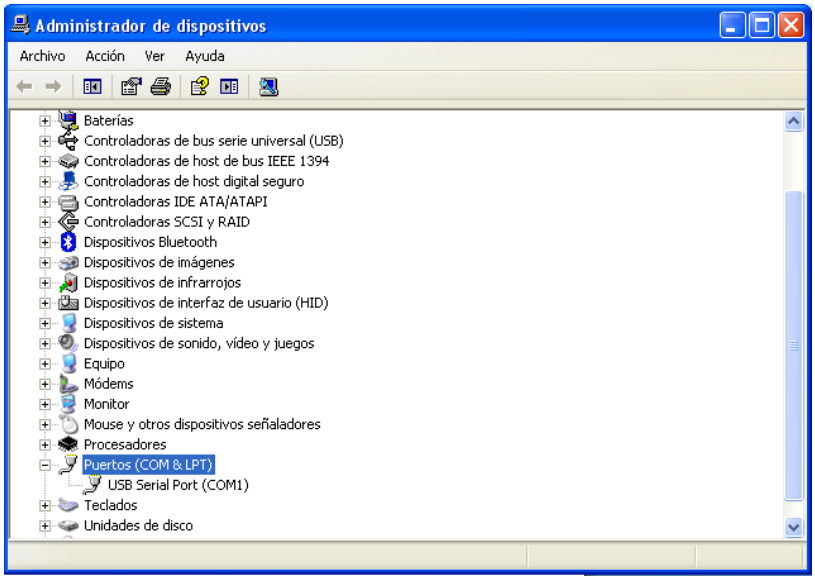
Este mensaje de error se produce cuando no hay conectado ningún puerto en la cámara seleccionada o el nombre del puerto no está definido correctamente.

Solución: Renombraremos todos los puertos serie y introduciremos la configuración adecuada para cada puerto.

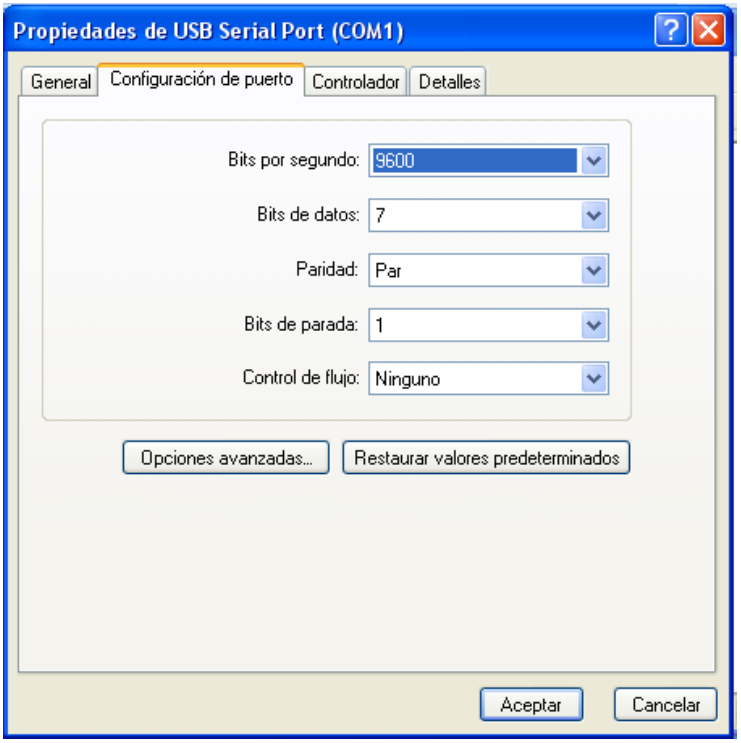
Nota: El software tiene por definición el puerto COM1, por lo tanto para que funcione correctamente el software tiene que haber uno de los puertos del PC establecido en ‘COM1’.

- Abrimos el administrador de dispositivos: **Panel de Control -> Sistema -> Hardware -> Administrador de Dispositivos.**

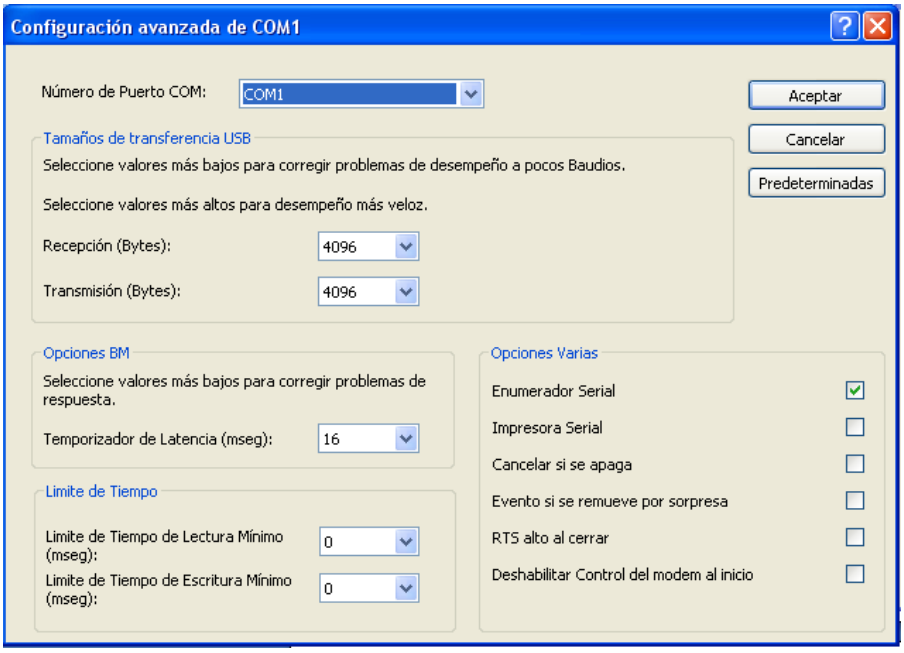




- Hacemos doble Click sobre el dispositivo serie ‘USB Serial Port (COM1)’ y seleccionamos la pestaña ‘**Configuración de puerto**’.

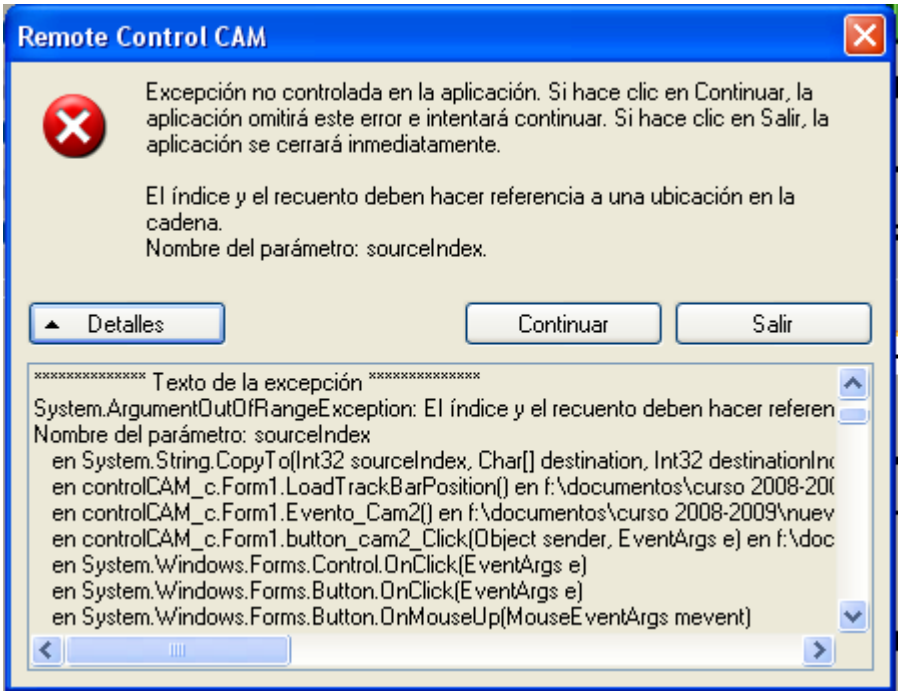


- Presionamos el botón ‘Opciones Avanzadas…’ y cambiamos el numero de puerto serie por el número de puerto establecido en la ‘Configuración SerialPort’ del programa.



12.5.2. Mensaje de error: “Excepción no controlada en la aplicación”

Se produce cuando en uno de los archivos “.TXT” de LOAD/SAVE de las variables del software ha sido cargada incorrectamente o los archivos han sido borrados.



Solución: Copiamos los siguientes archivos “.TXT”, restaurando todos los valores de los controles de software.

Directorio de archivos de restauración:
..\Ficheros anexos\Software y cabeceras\COPIA DE SEGURIDAD_ARCHIVOS LOAD-SAVE