UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Estudio de la cámara inteligente VC4018 y programación mediante QT para la conexión con dispositivo móvil Nokia.

Autor: Eduardo Tío Bellido
Director : Ángel Rodas Jordá

Fecha: 27 de julio de 2012

INDICE:

1-Motivación y objetivos.

Cada día crece el número aplicaciones, procesos y lugares que requieren la presencia de una cámara de video que haga las funciones correspondientes para el beneficio de las personas que las demandan ya sea en procesos industriales, tareas en las que sería muy peligrosa la presencia humana, simple vigilancia o procesos complejos en los que el ojo humano sería impensable que actuase debido a sus limitaciones.

Las cámaras de video o fotografía digitales unidas con el extenso abanico de tecnología ampliamente desarrollada existente en el mercado: ordenadores móviles, procesadores, sensores etc., ofrecen infinitas y muy sofisticadas aplicaciones de uso; haciendo posible una gran cantidad de tareas impensables en el pasado convirtiéndose en muchas ocasiones en un ojo humano de altas prestaciones.

Por su similitud con nuestro sentido de la vista, y la gran cantidad de información que con el podemos adquirir, la importancia que para nosotros representa el tener esa gran posibilidad de percibir el entorno que nos rodea, es el motivo por el cual haré un estudio de una de las cámaras inteligentes que el laboratorio de Visión por Computador del departamento DSICA de la Universidad Politécnica de Valencia tiene a disposición de los alumnos, con el objetivo de entender su funcionamiento, descubrir sus posibilidades y poner a prueba los conocimientos de informática adquiridos durante la carrera.

La visión es una forma muy fácil y rápida de obtener información del entorno que rodea un fenómeno, por eso en las aplicaciones que intentan imitar al comportamiento humano o convertirse en una extensión de sus acciones introducimos cada vez más y más cámaras que actúan como ojos sin descanso de una forma cotidiana.

Imagen y tecnología son motivo de estudio en las universidades debido a su gran número de posibles combinaciones ya sea en avanzados proyectos aeroespaciales o para hacer una simple foto de recuerdo con el móvil.

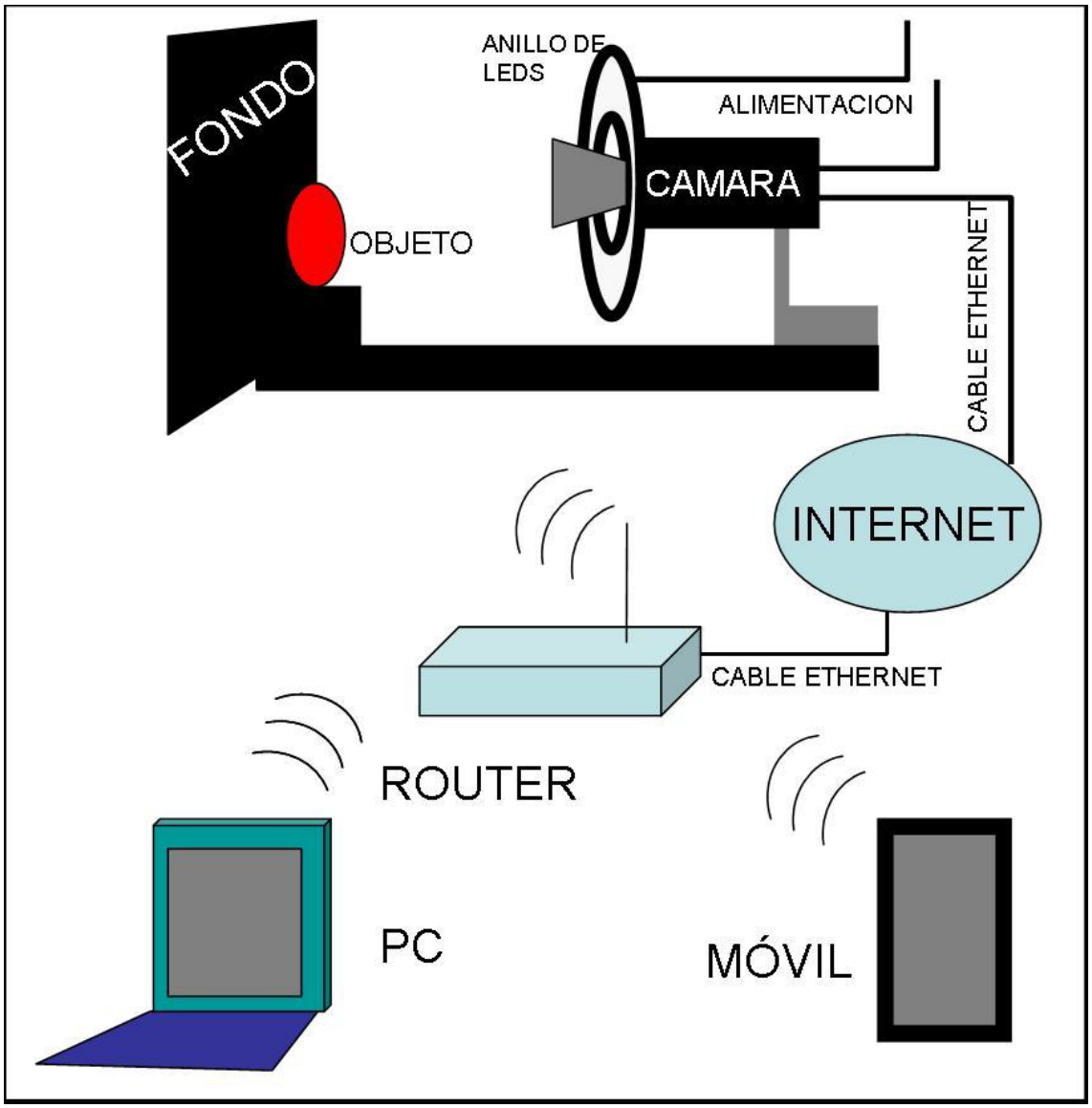
Por esas razones aprenderemos el funcionamiento de una cámara digital, como forma una simple imagen transmitiéndola o simplemente almacenándola en su memoria; el partido que podemos sacar al procesador de dicha cámara y como utilizarla con algunos dispositivos del mercado como pc y teléfono móvil etc.

La cámara con la que vamos a trabajar será modelo VC4018 de Vision Components, Un computador con todo el software necesario para la programación de la misma, acceso a internet, un portátil personal con todo el software de QT para programar el móvil, routers, conectores y cables etc, también contaremos con una gran documentación en ingles del fabricante Vision Components sobre la cámara VC4018, herramientas todas ellas muy interesantes para la realización de un buen y completo proyecto de vanguardia y muchas posibilidades.

Vamos a conectar la cámara al PC a través de internet para su configuración y uso debido a que es un proceso más rápido y ofrece muchas más opciones a la hora de comunicarse con otros dispositivos, ello posibilita su acceso desde cualquier parte del

mundo con conexión a la red; también usaremos un Router para conectar el PC y el móvil a internet.

La cámara dispone de iluminación necesaria para una correcta visión del objeto que se sitúa a una distancia de trabajo adecuada según las características de la óptica. También disponemos de sensor laser que podremos utilizar para disparo de captura de imagen, será comentado como curiosidad y ejemplo de uso. Tendremos soportes para su cómoda sujeción. El esquema que se muestra a continuación es el utilizado en el montaje:



Una vez configurada la cámara, el móvil y el PC podrán conectarse a ella mediante el Router con conexión a internet, el PC dispondrá de software específico para realizar operaciones con el dispositivo y mediante QT haremos una versión de dicho software

para utilizar en el móvil, permitiéndonos hacer con la cámara prácticamente las mismas acciones que con el pc.

OBJETIVO:

Conocer y comprender las características de una cámara inteligente junto con su entorno de programación y formas de interactuar entre dispositivos Diseñar sencillos programas y aplicaciones que permitan el procesamiento de imagen y el intercambio de información con diferentes dispositivos como PC's, móviles, etc.

Estudio y programación de dispositivo móvil mediante el entorno de trabajo QT

2-La cámara inteligente

2.1) Introducción

Cámara inteligente o smart camera en ingles es un dispositivo integrado orientado a la visión artificial. Las cámaras inteligentes poseen, aparte de una electrónica de captura de la imagen, un procesador con el cual tratar la imagen capturada sin necesidad de una CPU externa.

Suelen poseer también conexiones de tipo Ethernet o puerto serie, entradas y salidas digitales (E/S), líneas de control y potencia de iluminación, etc.

Si bien la potencia de una cámara inteligente no consigue los resultados en rendimiento de un sistema de visión artificial basado en PC, sí se acerca mucho en cuanto a las posibilidades del software de tratamiento de la imagen.

El éxito de las cámaras inteligentes se debe principalmente a sus reducidos tamaño y coste

Las cámaras inteligentes son cámaras digitales, una cámara digital es una cámara fotográfica que, en vez de captar y almacenar fotografías en películas química como las cámaras fotográficas de película fotográfica, aprovecha el proceso de la fotografía digital para generar y almacenar imágenes.

Las cámaras digitales compactas modernas generalmente son multifuncionales y contienen algunos dispositivos capaces de grabar sonido y/o video además de fotografías

Las cámaras digitales suelen emplear la óptica y los mecanismos de las cámaras tradicionales, pero sustituyen la película por un foto-sensor electrónico (CCD, CMD o Cmos).

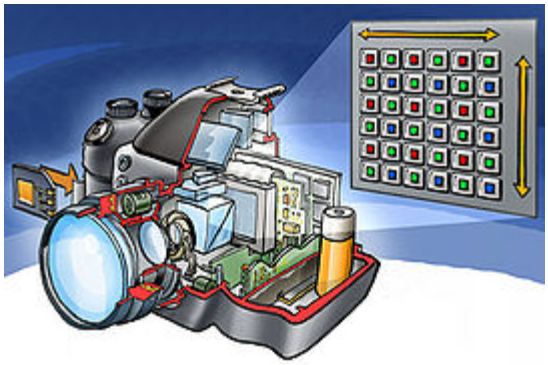
Generalmente, el foto-sensor es un CCD de tipo área (Area Array CCD), consistente en una matriz reticular de cientos de miles de células fotosensibles microscópicas (fotodiodos). A cada fotodiodo le corresponde un píxel, por lo que cuantos más foto-sensores tenga el CCD, mejor será la calidad obtenida con la cámara, siendo valores

habituales en las cámaras actuales 128.000 (320 x 400 píxeles de resolución) en las de gama baja, 4.200.000 (2.024 x 2.024 píxeles) en las de gama media y más de 6.000.000 en las profesionales de gama alta.

El proceso de captura puede realizarse en una o más pasadas, en cada una de las cuales se recoge información del modelo real. En caso de captura en una sola pasada, uno de cada cuatro elementos del CCD lee la información correspondiente al rojo, otro la correspondiente al verde y los dos restantes la correspondiente al azul, siendo rellenados los vacíos de información cromática que se produzcan mediante interpolación.

La captura puede hacerse también en método entrelazado, en el que el sensor de la cámara recoge información sobre la imagen procesando primero las líneas impares y luego las pares, o en el método progresivo, en el que el sensor recoge información sobre la imagen procesando las líneas de forma secuencial, una detrás de otra.

Una vez capturada la imagen, es necesario almacenarla temporalmente en la cámara hasta su descarga al ordenador. Dependiendo de la marca y del modelo de la cámara se guardará la imagen digital en formatos gráficos puros, como RAW, TIF, FlashPix o Targa (TGA).



Esquema de cómo se almacena la información de la luz en las celdas de los sensores, cada celda es un pixel.

En una imagen o mapa de bits cada pixel es codificado con un conjunto de bits, esto se llama profundidad del color, cada pixel puede codificarse con un byte (*que consta de 8 bits*) esto da un número de 256 variaciones.

En una imagen de color verdadero (*true color*) se usan tres bytes para definir un color, equivalentes a un valor tonal de 224 colores, cuyo resultado es 16.777.216 opciones de colores diferentes. La diferencia entre esta profundidad de color de 24 bits y una de 32, son 8 bits más para un canal de transparencia o alfa.

En nuestro caso la imagen capturada por la cámara tendrá un byte por pixel, la imagen será en escala de grises.

2.2) La cámara VC-4018

El modelo de cámara utilizado será VC4018, en la siguiente imagen se puede observar la estructura de la cámara VC4018E y los interfaces de conexión (se puede apreciar su construcción robusta para poder resistir las vibraciones y golpes).

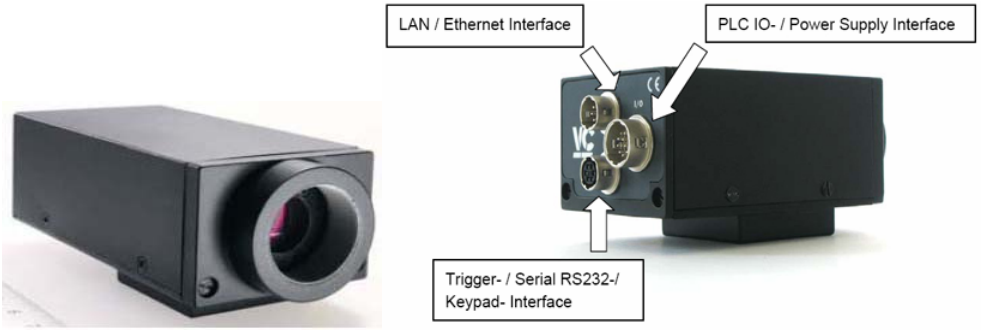


Imagen 2.1 Modelo **VC4018E**

Este modelo de cámara tiene los siguientes interfaces de conexión.
• Interface de alimentación PLC IO.
• Interface LAN / puerto serie estándar.
• Interface serie 24V Trigger- Equipad

Nota: Este modelo de cámara no tiene salida de video. El video lo podemos visualizar en el PC, conectando la cámara directa a PC o conectándola a la Ethernet. En próximos puntos de este informe se explica el procedimiento.
En la tabla 2.2, se pueden observar las principales características técnicas del modelo de cámara **VC4018E**.

Component / Feature	Specification
CCD Sensor:	1/3" SONY ICX424AL - also available with color sensor (Bayer Filter)
eff. no. of pixels:	640(H) x 480(V)
Pixel size:	7.4(H) x 7.4(V) µm
Chip size:	5.79(H) x 4.89(V) mm
High-speed shutter:	36.2, 98.6, 161 microseconds, increasing with steps of 62.4 microseconds (full-frame shutter)
Low-speed shutter:	up to 2 sec. adjustable integration time
Integration:	full-frame
Picture taking:	program-controlled, trigger controlled (interrupt); full-frame / 32 frames per second, external high speed trigger
Clamping:	zero offset digital clamping
A/D conversion:	12.5 MHz / 10 bit, only the 8 most significant bits used for grey values
Input LUT	none
Image Display	Via 100 Mbit Ethernet onto PC
Processor:	Texas Instruments 400 MHz TMS320 C64 DSP
RAM:	32 Mbytes SDRAM (synchronous dynamic RAM)
Memory capacity:	Up to 100 full-size images in format 640x480
Flash EPROM:	4 Mbytes flash EPROM (nonvolatile memory) for programs and data, in-system programmable, 3 MB available to user
MMC:	Not available
Process interface:	4 inputs / 4 outputs, outputs 4x400 mA
Trigger Input	Fast 5 V TTL input and output, jitter free image acquisition
Serial Interface:	115,200 bd serial RS232 communication port
Ethernet interface:	100 Mbit
Video output	No direct video output / download of live images via Ethernet possible
CE certification:	CE Certification from Vision Components
Storage Conditions	Temperature: -20 to 60 deg C, Max. humidity: 90%, non condensing.
Operating Conditions	Temperature: 0… +50 deg C (housing temperature), Max. humidity: 80%, non condensing.
Power Supply	12V… 24V
Power Consumption	≈3 W (current drawn from PLC outputs additional)

Nota: En la imagen 2.1 se puede observar que la cámara no tiene la óptica insertada.

La cámara utiliza el procesador TMS320C6424 de Texas Instruments, cuyo diagrama de bloques es el siguiente:

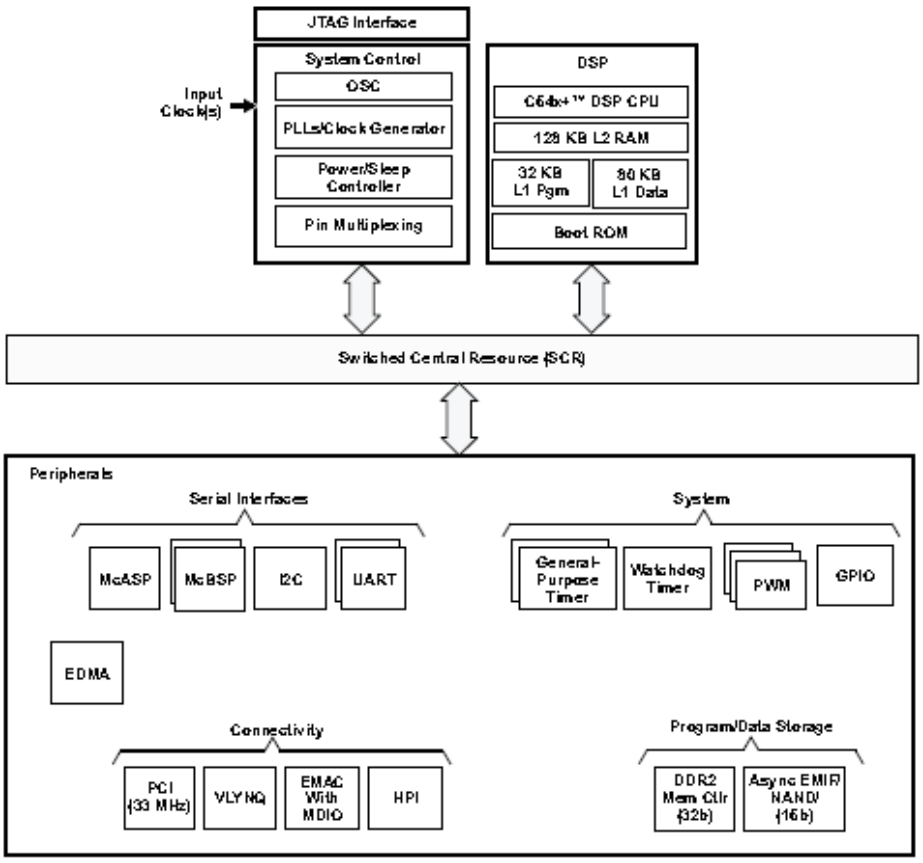
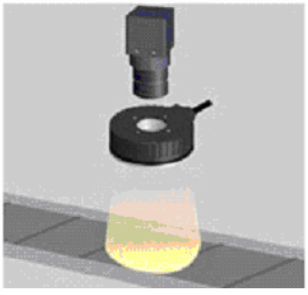


Figure 1-1. TMS320C6424 Functional Block Diagram

En Próximos puntos de este informe veremos el tipo de ópticas utilizadas.
La iluminación que se ha utilizado durante el montaje, ha sido:

- Anillo de iluminación.
- Regleta de focos.
- Anillos de Leds.

Estos últimos son los que se han montado, a la finalización del proyecto. Se hace referencia a los anillos de LED´s.
Los sistemas anulares son sistemas anulares de iluminación formados por un anillo de LEDs, que permite la colocación de la cámara en el orificio formado por el anillo.



Colocación de la cámara.

Esta técnica de iluminación es la más común. La cámara se posiciona mirando al objeto en la misma dirección que la luz. La cámara recibe la luz reflejada del objeto. Se suministran en diferentes intensidades, longitudes de ondas y medidas.
Esta iluminación es especialmente útil en superficies con pocos reflejos (papel, tela,…). Existen sistemas anulares de diferentes dimensiones y en distintas longitudes de onda (Blanco, IR, Rojo, Verde y Azul).

Nota 1: LED rojo la alimentación es de 12V y en el caso de los Blancos, Azules y Verdes es de 24V.

Las ópticas están disponibles en distintas distancias focales desde 3.5 a 200mm. Estas ópticas están optimizadas para enfocar desde infinito hasta pocos centímetros del objeto.

Son ideales para la mayoría de cámaras estándar. Este tipo de ópticas diseñadas para aplicaciones industriales utilizan cristales de calidad y son de construcción robusta, para poder resistir las vibraciones y golpes sin que se desenfoque o cambie la apertura. Están provistas de tornillos de fijación para el iris y el enfoque.
Para saber exactamente que óptica debe utilizarse para la aplicación que se desea resolver se debe tener en cuenta una serie de parámetros. Por una parte se debe conocer el tamaño del sensor de la cámara, también se debe saber a que distancia estará el objeto de la cámara y por último se debe conocer el campo de visión que deseamos abarcar en nuestra aplicación. Una vez conocidos todos estos parámetros podremos calcular la óptica a utilizar mediante la siguiente fórmula:
Distancia = (Tamaño del sensor * Distancia al objeto) / Tamaño del objeto.

En la tabla 2.5 se pueden ver las características de las ópticas. Se marca la óptica que mejor se adapta a las exigencias del sistema.

Modelo	Fabricante	Distancia focal (mm)	Apertura minima (mm)	Distancia (mm)	Formato	Rosca filtro	Ø	Longitud (mm)
OPT-M23514MCN	INFAIMON	3.5	1.4	200	½	NO	31	30.5
OPT-M24514MCN	INFAIMON	4.5	1.4	200	½	NO	31	29.5
OPT-M26014MCN	INFAIMON	6	1.4	200	½	27.0-0.5	30	30
OPT-M38014MCN-1	INFAIMON	8	1.4	100	2/3	25.0-0.5	30	30
OPT-M31214MCN-1	INFAIMON	12	1.4	100	2/3	27.0-0.5	30	31.5
OPT-M31614MCN-1	INFAIMON	16	1.4	200	2/3	27.0-0.5	30	28
OPT-M32516MCN-1	INFAIMON	25	1.6	200	2/3	27.0-0.5	30	28.5
OPT-M33516MCN-1	INFAIMON	35	1.6	350	2/3	30.5-0.5	32	36.5
OPT-M35020MCN-1	INFAIMON	50	2.0	500	2/3	30.5-0.5	32	40
OPT-M37525MCN-1	INFAIMON	75	2.5	1200	2/3	34.0-0.5	36	51
OPT-M310028MCN-1	INFAIMON	100	2.8	2000	2/3	40.5-0.5	42	70

Tabla 2.5. Tabla de ópticas estándar

Comunicación PC cámara.
Esta comunicación **(comunicación directa)**, la utilizaremos para la configuración de las IP´s de las cámaras, como también para la extracción de la MAC de las cámaras. La IP por defecto de la cámara es la siguiente: 192.168.0.65, para el modelo VC40XX.

Comunicación PC ethernet cámara
Esta comunicación **(comunicación múltiple)** es la utilizada en este proyecto. De esta forma se puede acceder a mas de una cámara desde el propio PC, así como desde otro PC del Laboratorio o exterior a él. Al acceder a las cámaras se puede realizar una comunicación simultánea.
Las cámaras vienen con una fuente de alimentación cada una. Se hace referencia a los pines de alimentación de cada cámara en sus correspondientes manuales.

Programación del sistema.

LOS COMANDOS “SHELL”.
Con los comandos Shell actuales (similares a los comandos MS-DOS, y el prompt es: $), solo se pueden acceder a través de una interfaz estándar. En la tabla 5.1 se

describen las funciones de los principales comandos, permiten controlar la cámara, a través de comandos de teclado.

Function	Keyboard Command
Uploading of programs from PC in to camera flash memory	$lo (and then activating the upload function of the terminal program).
Program execution	$myprogram
Display the File Directory on fd (md)	$dir ($dir md:/)
Delete Files (from directory content)	$del myfile
Packing the eprom (freeing memory of deleted files)	$pk
Turn video output to live (still) mode	$vd –l (vd –d)
Check Camera OS version	$ver
Display ASCII file content	$type myASCIIfile (e.g. #ID to display camera ID)

Tabla 5.1. Principales funciones

La cámara puede manejarse fácilmente mediante estos comandos muchos de los cuales son sobradamente conocidos.

Como por ejemplo *Dir, Copy, Help*, utilizados en los principios de los ordenadores personales los cuales carecían de interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador, los primeros GUI aparecieron en los años 80.

En la siguiente figura se describen los siguientes comandos internos (en orden alfabético):

bd	set baud rate	bd <baudrate>
cd	change data directory	cd <path>
cx	change execution directory	cx <path>
copy	copy a file	copy <source path> [<dest path>]
del	delete file	del <path>
dir	directory of Files	dir [<option>] [<path>]
disp	**switch display modes**	**disp [<option>] [<mode>]**
dd	DMEM Display	dd <addr>||<range>
dwn	download file to PC	dwn <path>
er	erase complete flash eprom	er
ex	exit from shell	ex
fmt	**format media card**	**fmt [<size> [<clustersize>]]**
?	help	? [<name>]
he	help	he [<name>]
help	help	help [<name>]
ht	hardware test	ht
js	jpeg store	js <path>
jl	jpeg load	jl <path>
kill	**delete task**	**kill <PID>**
kl	**kernel log**	**kl**
lo	load S records	lo
mdir	**display module directory**	**mdir [<option>]**
mem	**display memory usage**	**mem [<option>] [<PID>]**
mkdir	make directory	mkdir <path>
ping	**test IP connection**	**ping <IP-address>**
pk	pack flash memory	pk
procs	print task list	procs
sh	set shutter value	sh <number>
time	time and date command	time [<option>]
tp	take picture	tp
type	type ASCII file	type <path>
ver	print software version	ver
vd	video modes	vd [[<option>] <frame number>]
wb	**whitebalance**	**wb**

Figura 5.1. Descripción de los comandos internos

Estructura de los directorios y Tipos de archivos soportados.
En la Tabla 5.2 se resumen los directorios y archivos que soportan los modelos de cámara VC40XX.

Flash EPROM	MMC / SD
- Does not support standard file extensions! The following files types are supported and automatically converted into numerical file extensions (FTP transfer): - exe → 000 - txt → 001 (also: asc, htm, html) - dat → 002 - jpg → 003 - binary → 231 This extension is also displayed if one of the file types above is not recognized (adjust file extensions on PC in this case).	- Does support standard file extensions
- EPROM required packing in order to free memory after file deletion (see 10.1).	- MMC and SD are self organizing, i.e. packing after deletion not required.
- it is not possible to create subdirectories	- Creation of subdirectories possible - $mkdir md:/subdir
- Faster access	- Slower access
- Protected first flash EPROM sector: fd:/sys/ Accessible with "$dir –x" - Working directory: fd:/user	- No subdirectories - Path: md:/
- File upload per Telnet/ RS232 /FTP possible	- File upload only per FTP possible
- Only *.msf file upload (see section 10.3.1)	- Upload of standard file types (no *.msf)
- Default directory for program execution	- Executing programs from MMC possible after calling : $cx md: (back: $cx fd:)

Tabla 5.2. Directorios y archivos soportados

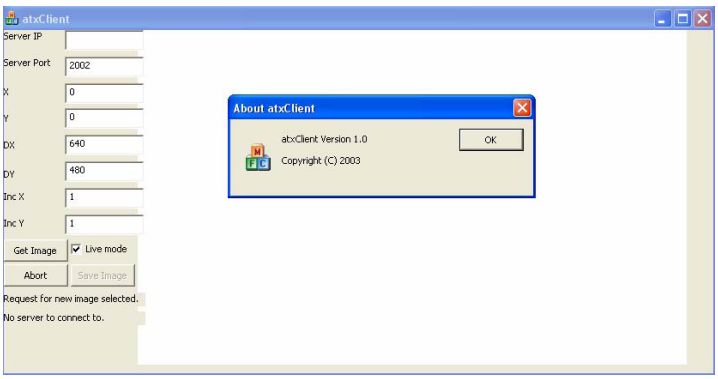
COMUNICACIÓN CON LA CÁMARA.
El modelo de cámara VC40XX incorpora una interfaz estándar.
- Ethernet TCP / IP Telnet puerto 23 'comandos' y por el puerto 2002 'datos', para la cámara VC40XXE.

A través del terminal de programa Tera Term.
La comunicación de la cámara se realiza con el programa TeraTerm.
Tera Term (Pro) es un emulador de terminal de software libre (programa de comunicación para Windows por el puerto 23). Es compatible con la emulación VT100, conexión telnet, y la conexión de puerto serie.

Entorno Tera Term

Se ha trabajado con la versión 3.1. Lo más utilizado ha sido a la hora de asignar una nueva conexión, así como la carga de nuevos archivos de programa para la cámara. Se pueden hacer referencia a diferentes configuraciones como:
- Guardar las diferentes conexiones IP.
- Configurar rutas de apertura de carpetas.
En este entorno se utilizan los comandos shell de la cámara.

AtxClient
Programa en el cual podemos visualizar las imágenes de la cámara en directo o capturar la imagen y guardarla en disco.



Entorno AtxClient

Funcionamiento.
Modo directo:
Insertamos en el campo"Server IP" el número IP correspondiente de la cámara, activamos el campo "Live Mode" y pinchamos en "Get Image" en la ventana adyacente, vemos la imagen en directo que nos llega de la cámara.

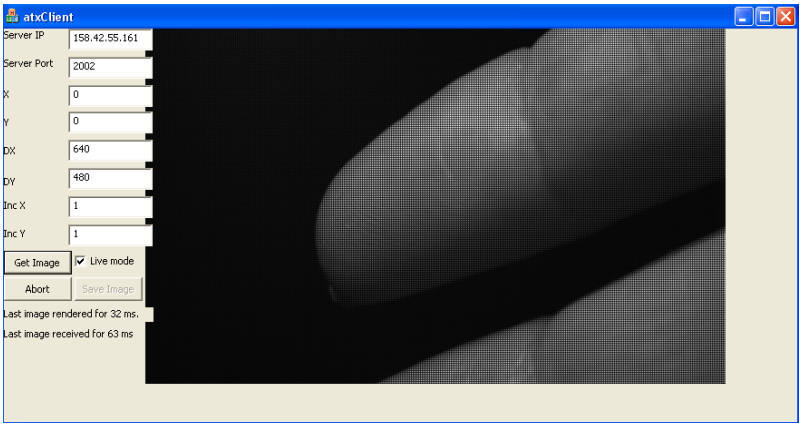


Imagen en modo live **VC4018E**.

Modo 'Guardar Imagen':
Realizando el proceso anterior "Modo directo" deshabilitamos el campo "Live mode", ahora estará habilitado "Save image", el cual pulsaremos para guardar la imagen que se ha quedado congelada en la ventana del entorno AtxClient.

Carga de programas utilizando el Tera Term.
Después de establecer la conexión con la cámara a través del Terminal de programa, los programas pueden ser cargados.El comando shell **"lo"** se utiliza para cargar los archivos ***. msf** en la cámara. Los siguientes pasos son necesarios para subir el archivo *. msf con TeraTerm:

Commands:	Explanation
1. "$lo"	Calling the shell command "lo" without parameter activates the file upload. Confirm with Return. The camera sends "start address for file: xxxxxx" and waits for activation of the terminal upload function (program specific).
2. Select "Send File" under the "File" menu and browse for the *.msf file to upload.	Terminal program specific. Refer to the VCRT documentation for upload with ProComm.

Paso para la carga de programas.

Nota: Los archivos ***.msf** están en la siguiente dirección: **"c:\ti\myprojects\send\"** que es donde el C.C.S deja el archivo ***.msf** después de realizar la compilación del programa en ***.c**.

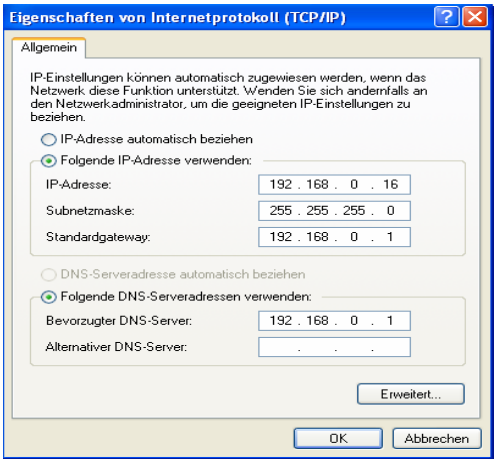
EXTRACCIÓN MAC, CÁMARA VC4018E.

1. **Conectar la cámara directamente al PC.**

Conexión directa PC cámara

Una vez desconectado el cable de red y conectado el cable de la cámara hay que configurar la nueva conexión de área local. Por defecto la cámara tiene la siguiente IP 192.168.0.65. Por ejemplo estas podrían ser las propiedades del TCP/IP del PC.

2. Comprobación



La comprobación la he realizado de la siguiente forma:
Abrimos la ventana de símbolo de sistema, y escribimos en la línea de comandos, he ping y la dirección IP de la cámara, este ha sido el resultado:

C:\>ping 192.168.0.65
Haciendo ping a 192.168.0.65 con 32 bytes de datos:
Respuesta desde 192.168.0.65: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.65: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.65: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.65: bytes=32 tiempo<1m TTL=64

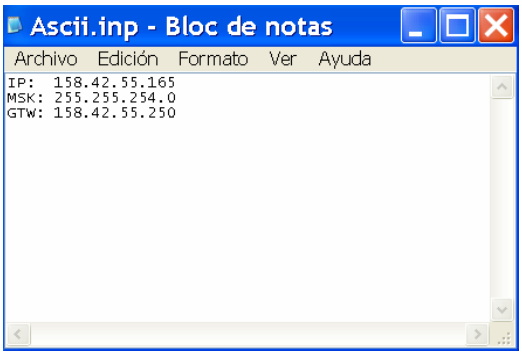
Estadísticas de ping para 192.168.0.65: Paquetes: enviados = 4, recibidos = 4, perdidos = 0(0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
Mínimo = 0ms, Máximo = 0ms, Media = 0ms

CONFIGURACIÓN IP CÁMARA.

La dirección IP de la cámara se cambia mediante la subida de un archivo en la memoria de la cámara, con la nueva IP. Este es un archivo ASCII que contiene la configuración de la nueva dirección IP:
La dirección IP por defecto, de los modelos de cámara VC40XX es 192.168.0.65.
La nueva IP de la cámara será 158.42.55.161.

Modificar la dirección IP de la cámara se realiza siguiendo los siguientes pasos:

1. Editar el archivo ASCII.INP.
En la siguiente ruta C:\ti\myprojects\demofiles VCRT5.00\ Ethernet New IP, ahí se encuentra el archivo ASCII.INP. Abrimos con el bloc de notas ASCII.INP.
2. Escriba la nueva dirección IP. Editamos los campos IP, MSK, GTW y guardamos.



Edición de archivo Ascii.inp

3. Haga doble clic en el archivo cc_IP.bat. Este archivo cc_IP.bat genera el ip.msf.

4. Cargar el archivo ip.msf en la cámara. Conectamos la cámara directamente a PC.

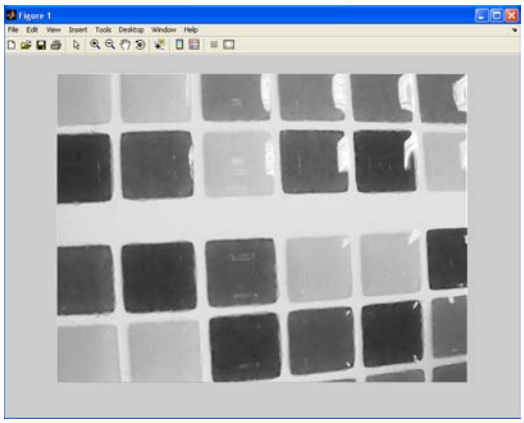
Abrimos la consola **Tera Term**, creamos la conexión con la cámara con la IP por defecto.
Presionamos la tecla **"enter"** ya que es el password de acceso a la cámara (si no se ha cambiado). Escribimos en la línea de comandos **"lo"** pinchamos en **File>Send File** y cargamos el fichero **ip.msf**. Para comprobar si el archivo se ha cargado escribimos en la línea de comandos **"dir"** y vemos todo lo que hay en memoria de la cámara. Una vez comprobado, deshacemos y volvemos a los valores que corresponden para conectar los dispositivos a la red. La cámara se tendrá que reiniciar, así esta tomará el valor de la nueva IP.
Para comprobar la conexión de red Podemos realizar un ping <dir IP> desde la línea de comandos de Windows.

La cámara VC4018 puede obtener imágenes bayer o rgb, la distribución de los píxeles para el almacenamiento del patrón bayer, modelo de cámara VC4018/E. podemos verla en esta tabla.

R	G	R	G	R
G	B	G	B	G
R	G	R	G	R
G	B	G	B	G
R	G	R	G	R

Distribución de píxeles CCD VC4018/C

La diferencia entre una imagen bayer y una imagen rgb podemos verla en estas dos fotos.



Patrones bayer

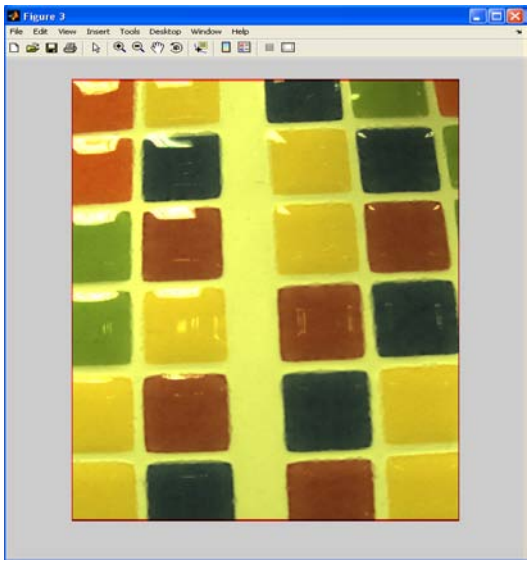


Imagen RGB

2.3) Entorno de trabajo

El escenario de trabajo consta de la cámara, superficie cuadriculada para calibrar objetos, soportes, cableado de red y para dispositivos opcionales como disparadores motores, sensores, etc y alimentación tanto de la cámara como de los anillos LED para iluminación, Router (en el caso de conexión con móvil) y barrera laser opcional.

El escenario quedará montado de la siguiente manera, con la cámara perpendicular apuntando a un fondo negro para resaltar objetos claros que son los que utilizaremos y

a una distancia suficiente de este marcada por las características de las lentes utilizadas.

Imagen del escenario con la cámara

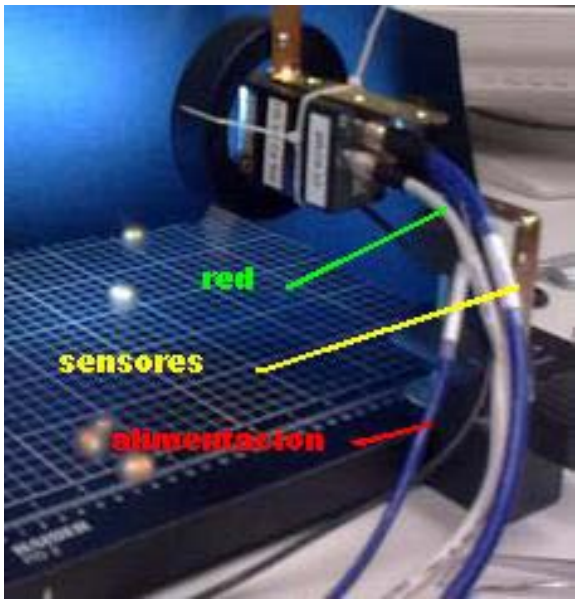


Imagen cableado

CONEXIÓN BARRERA LÁSER.

Una de las utilidades de la barrera láser puede ser el disparo de la cámara por la detección de un objeto, esto se produciría cuando el objeto interrumpe el haz de luz de la barrera.

Para el disparo de la cámara seguimos el siguiente esquema.

Conexionado para la VC4018E.

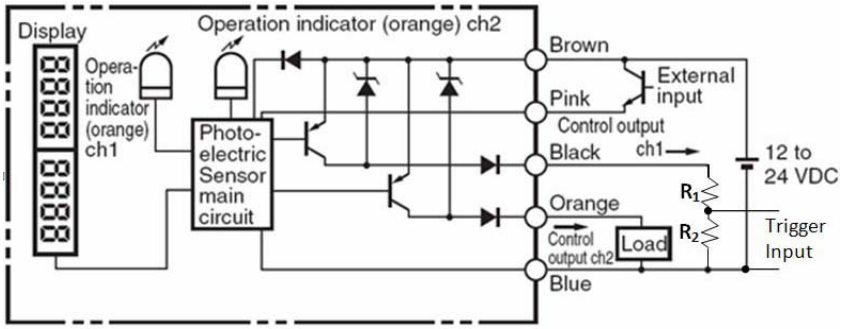


Imagen. Circuito de conexión entradas salidas **VC4018E**.

Señal	N° Pin	Color Cable	Jack
V24 TxD Out	1	Verde	
+5V Out	2	Marrón	
GND	3	Blanco	
V24 RxD In	4	Rosa	
Trig.Out	5	Gris	
Trig In	6	Amarillo	

Asignación de pines **VC4018E**.

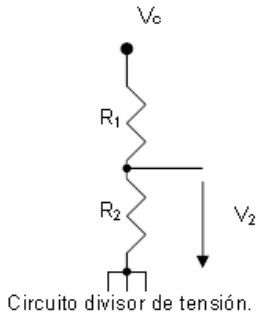
La alimentación del amplificador es de de 12-24 VDC. Con lo cual la tensión de salida del amplificador estará aproximadamente al valor de VDC que alimentemos.
El trigger de las cámaras, su valor de tensión de entrada es de 3-5VDC con lo cual tenemos que calcular un divisor de tensión a la salida del amplificador de fibra óptica.

Calculo del divisor de tensión

Valores teóricos del divisor de tensión.
Vcc= 12V
V2= 5V

$$V_2 = \frac{R_2}{R_1 + R_2} \times V_{cc}$$

Ecuación divisor de Tensión



Circuito divisor de tensión.

Valores teóricos.

$$5 = \frac{R_2}{R_1 + R_2} \times 12$$

Despejando y realizando las operaciones correspondientes queda:

$$R_1 = \frac{7}{5} R_2$$

Dando valores estándar (E12):

Le damos valor a R_2=1KΩ $\qquad R_1 = \frac{7}{5} 1 = 1.4k\Omega$

Valores reales a utilizar, según tabla estándar E12:

R_1=1,5KΩ
R_2=1KΩ

Comprobación de los valores que vamos a utilizar.

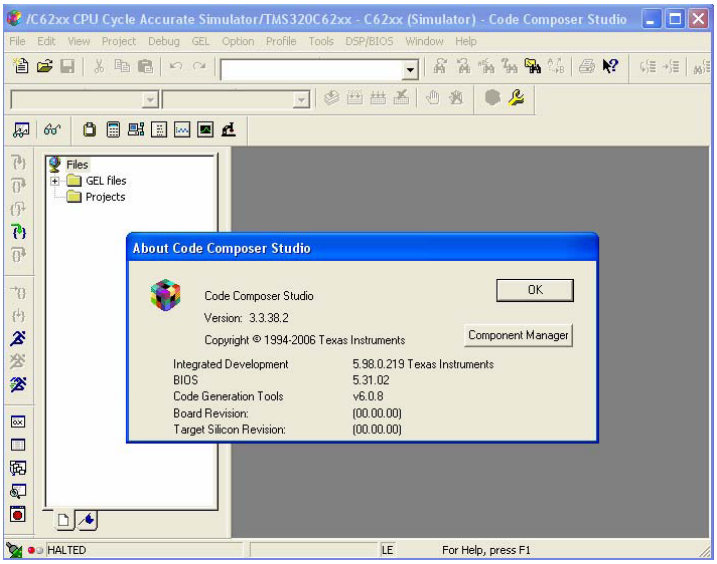
$$V_2 = \frac{1}{1.5 + 1} \times 12 = 4.8v$$

Estamos dentro del rango posible para que haya disparo de trigger. Recordamos que el rango de disparo de trigger es de 3-5V TTL.

3-Programación de la cámara y conectividad con el PC

3.1) Code Composer Studio

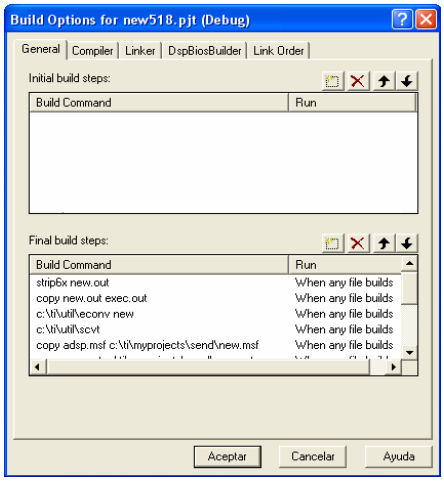
Entorno de trabajo del C.C.S v3.3 en el cual se desarrollan, modifican y compilan los programas en *.c.



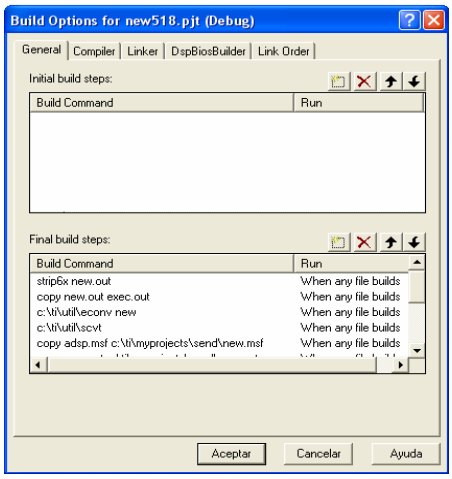
Entorno C.C.S v3.3

Del code Composer Studio destacamos la configuración de proyectos. Se describe los pasos a seguir para la configuración de un proyecto. Poject \ New Project (creamos el proyecto).
Entramos en el **“build options”** del nuestro proyecto:

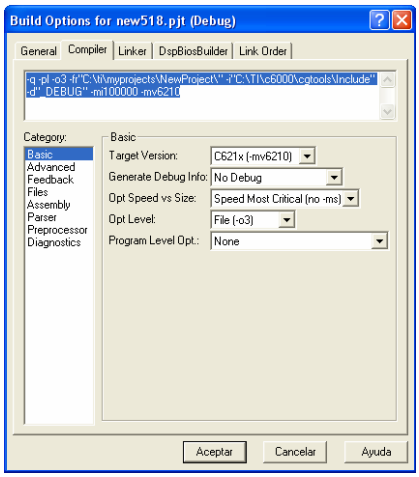
General



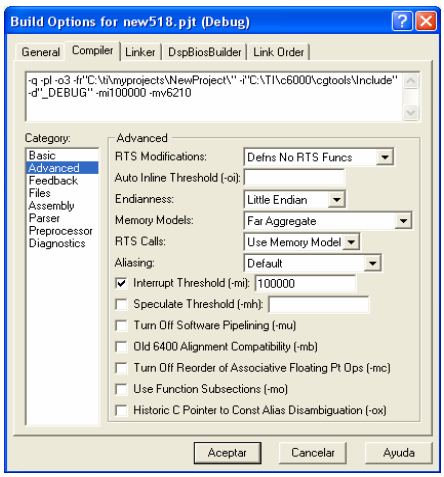
Compiler / Basic



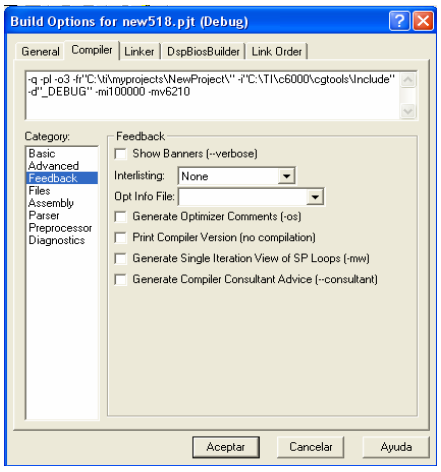
Compiler / Avanced

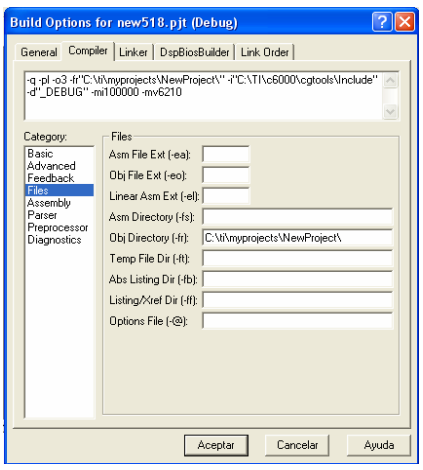


Compiler / Feedback

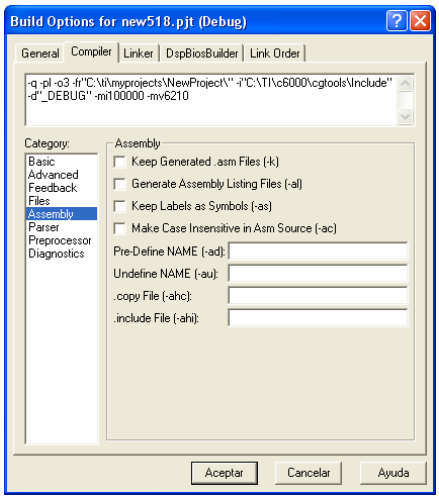


Compiler / Files

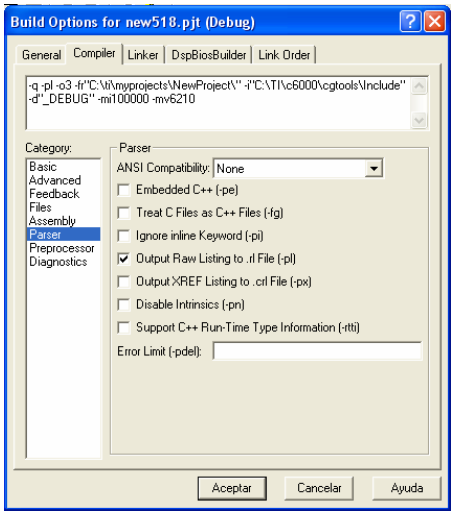




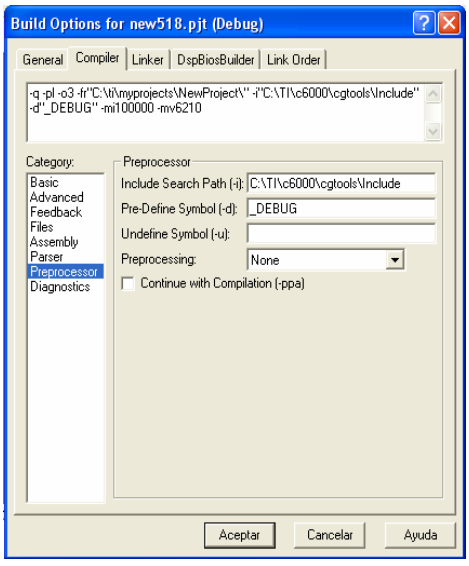
Compiler / Assembly



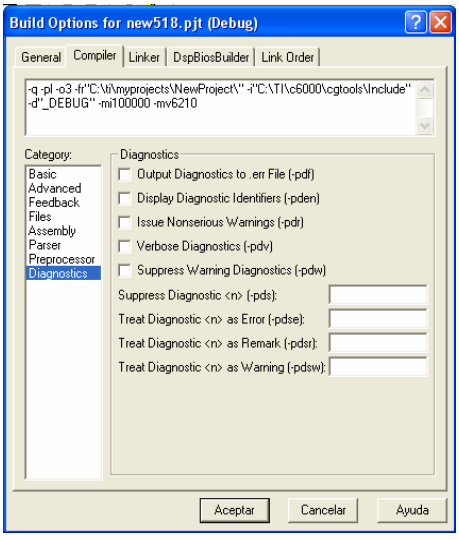
Compiler / Parser



Compiler / Preprocessor



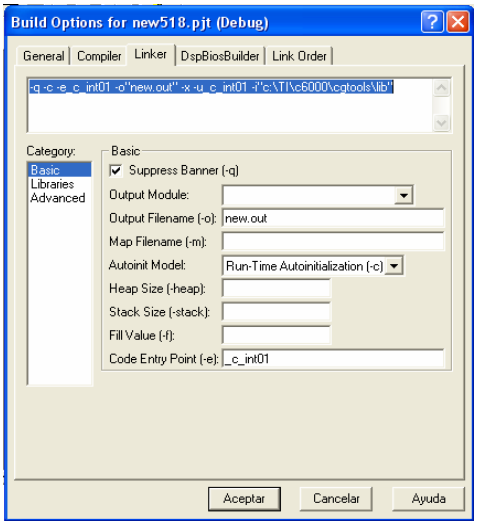
Compiler / Diagnostics



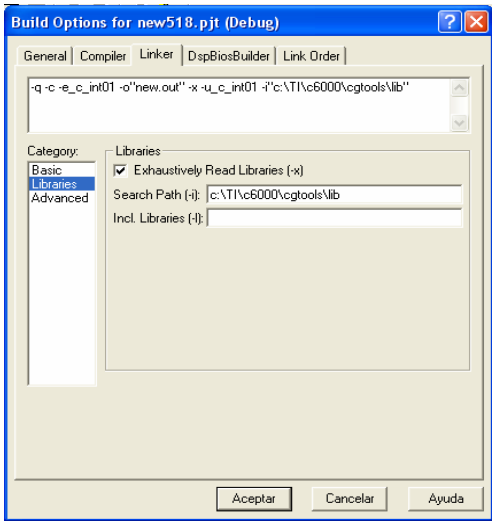
Aumentando las imágenes podemos ver lo que debemos escribir en el recuadro superior para la configuración de cada una de las etapas.

En esta configuración el Code Composer quedara listo para guardar los ficheros en la ruta indicada, podremos indicarle que programas utilizar para la transformación de los ficheros y con que nombre queremos que guarde el fichero final que enviaremos a la cámara mediante Tera Term.

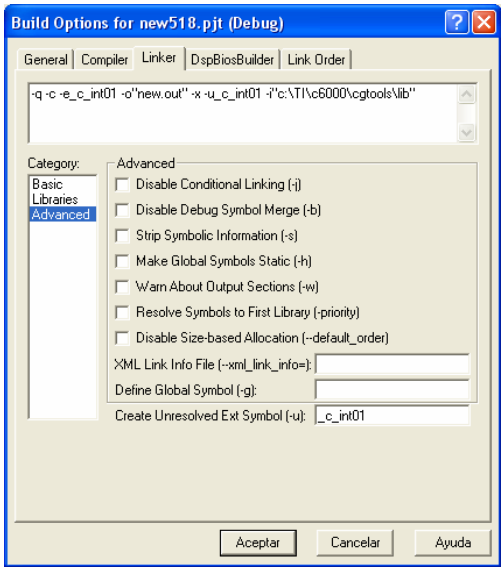
Linker / Basic



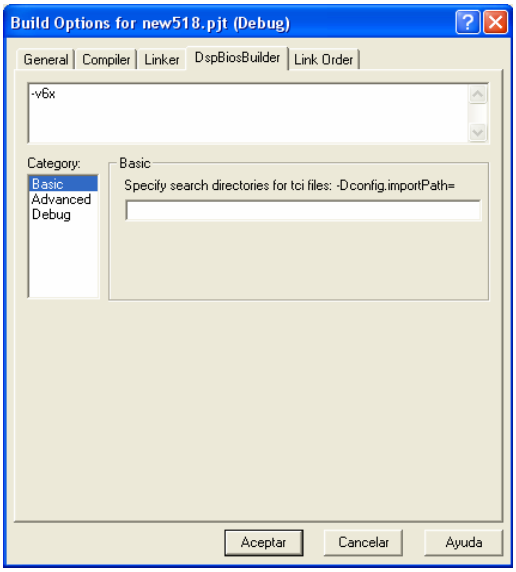
Linker / Libraries



Linker / Advanced

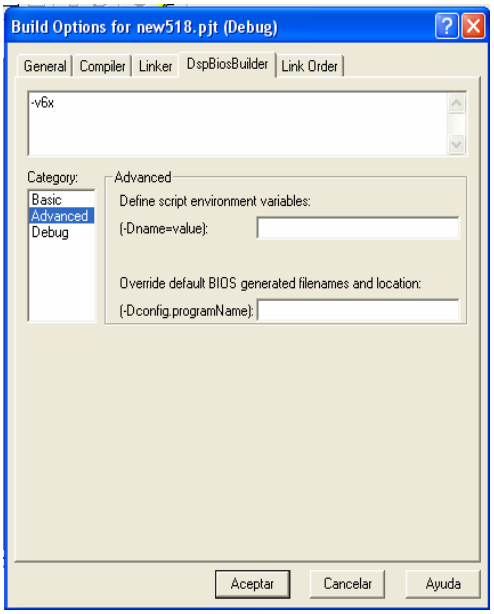


DspBiosBuilder / Basic

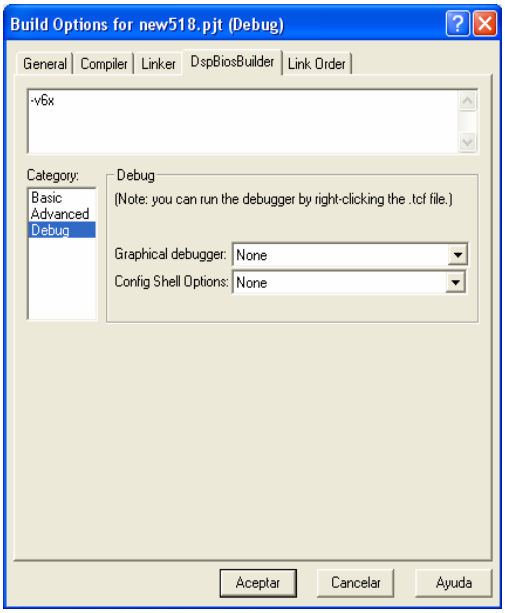


Indicaremos las librerías a utilizar, y configuraciones avanzadas las cuales deberemos seguir paso a paso este manual para un correcto funcionamiento.

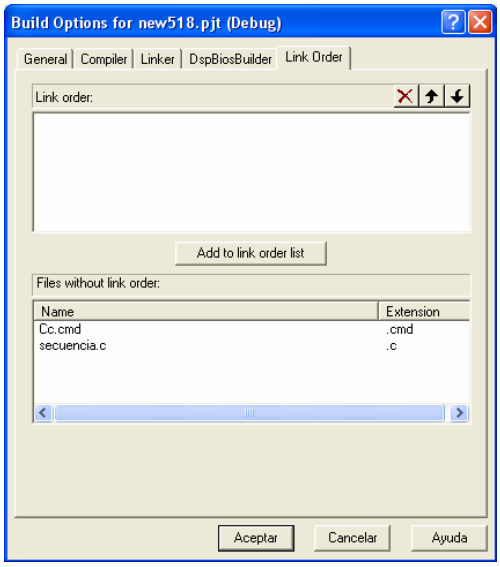
DspBiosBuilder / Advanced



DspBiosBuilder / Debug



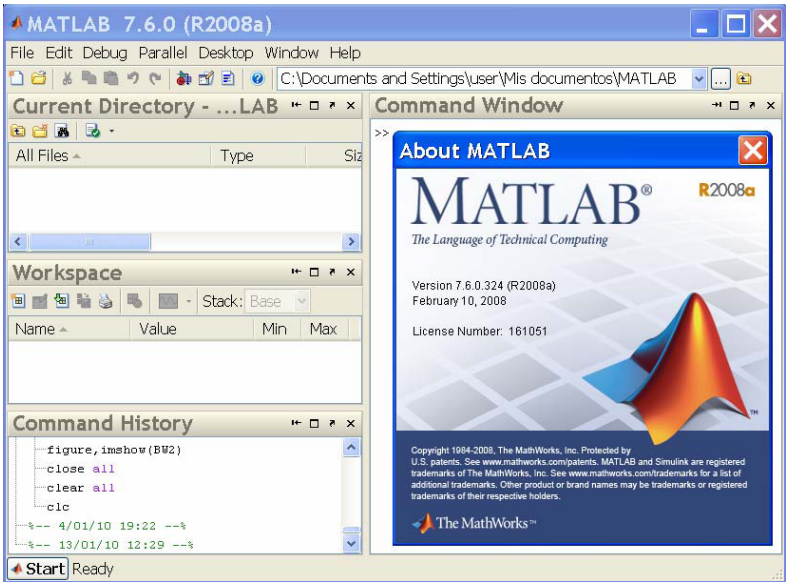
Link Order



Una vez configurado correctamente quedara listo para desarrollar proyectos correctamente.

3.2) Conectividad. Programación desde el entorno matlab

MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio.



Entorno Matlab

Algunos programas en Matlab para el uso de la cámara.

Comunicación PC cámara desde Matlab

```
while (isempty(s)) % entramos solo una vez en el bucle
con23=tcpip('158.42.55.165',23,'ByteOrder','littleEndian');%
Creamos conexión
fopen (con23);% abrimos conexión
respuesta= fscanf(con23,'%s',30);% Leemos los primeros 30
caracterés del puerto 23
s=strfind(respuesta,'password');% Buscamos la palabra ‘password’
if (~isempty(s)) % entramos solo una vez en el bucle
fprintf(con23,'%s','\n\nnew \n');% escribimos en puerto 23
‘new’(ejecutamos el programa cargado en memoria)
respuesta = fscanf(con23,'%s');%Leemos caracteres en
el puerto 23
end
end
```

Ejecución de programas de la cámara desde Matlab

Estas líneas de código equivalen, a la ejecución del programa (new) con el Tera Term.

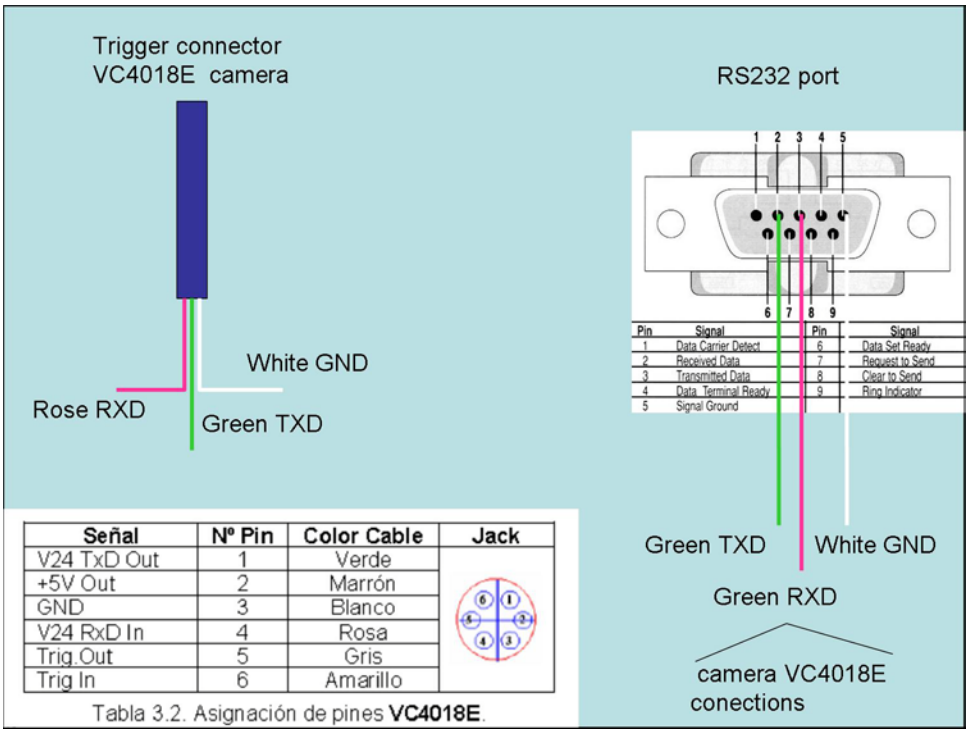
```
if (~isempty(s)) % entramos solo una vez en el bucle
fprintf(con23,'%s','\n\nnew \n');% escribimos en puerto 23
‘new’(ejecutamos el programa cargado en memoria)
respuesta = fscanf(con23,'%s');%Leemos caracteres en
el puerto 23
end
```

Blob.
Que realiza:

Este programa busca áreas negras en fondo un blanco y áreas blancas para un fondo negro. El programa los siguientes parámetros; el tamaño, centro de gravedad, número de objetos.

```
while (isempty(c))
datosascii=pnet(con23,'readline','intel'); % Lee una línea
completa
c=strfind(datosascii,'Analisis realizado en'); % Compara el
valor de datosascii con el patrón.
if isempty (c)
datosnum=str2num(datosascii); % Pasa de strig a numérico
fpritnf('Número de objetos:%d\n',datosnum);
end;
end;
```

Conexión por el puerto RS232, esquema conectores:



Señal	Nº Pin	Color Cable	Jack
V24 TxD Out	1	Verde	
+5V Out	2	Marrón	
GND	3	Blanco	
V24 RxD In	4	Rosa	
Trig.Out	5	Gris	
Trig In	6	Amarillo	

Tabla 3.2. Asignación de pines **VC4018E**.

3.3) Ejemplos de uso

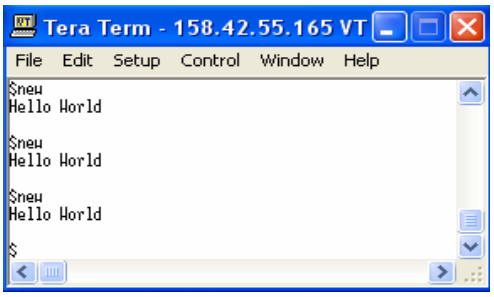
Vamos a ver algunos ejemplos de programas realizados con Code Composer cargados en la cámara y ejecutados mediante Tera Term.

Simulación con la consola TERA TERM.
Hello World.

Es el programa más sencillo para compilar el código fuente y cargarlo en la memoria de la cámara.

Que realiza:

Muestra en la consola del TERA TERM “Hello World”, realizando previamente la llamada al programa cargado en memoria.



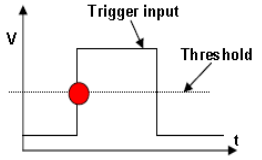
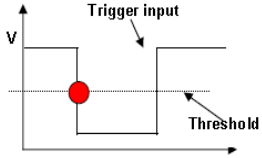
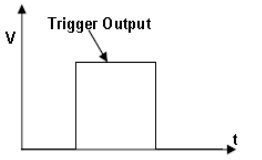
Resultados de la simulación

El código se puede ver a continuación:

```
/*********************************************************************/
#include <vcrt.h>
#include <vclib.h>                    // importación de librerías
#include <macros.h>
#include <sysvar.h>
/*********************************************************************/

void main(void)
{                                     //código
  print("Hello World\n");
  }
```

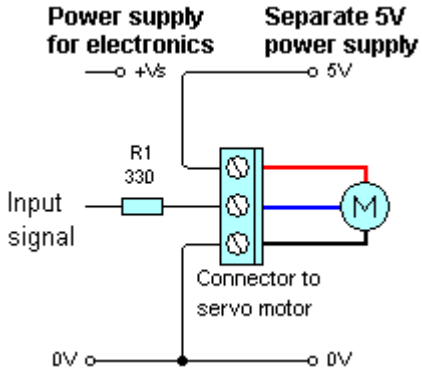
Aquí vemos las instrucciones que permiten poner el voltaje en la salida del trigger de la cámara a alto o bajo durante el tiempo deseado:

Signal	Trigger Mode
	TRIGINP_POS() Rising Edge Trigger Signal
	TRIGINP_NEG() Falling Edge Trigger Signal
	Trigger Output Signal in Exposure Controlled Mode: TRIGOUT_EXP(); AND Trigger output set positive: TRIGOUT_POS();

Con ello podemos controlar diversas aplicaciones como pueden ser el disparo de la cámara, un contador, un motor de pulsos etc, siempre con una salida de 5V.

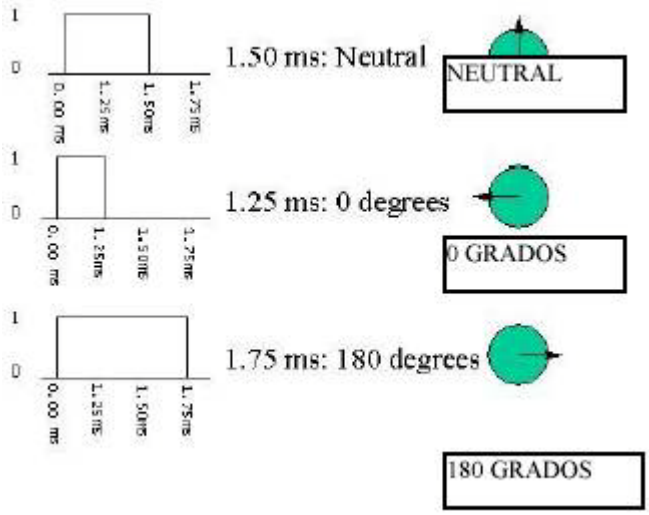
Aprovechando la salida del trigger podemos conectar un servo-motor que mueva la cámara con un programa cargado en ella, conectado la salida 5V (cable marrón de la

cámara) a la entrada 5V del motor, salida GND (cable blanco de la camara) y salida del trigger (cable gris de la camara) generando una señal a base de pulsos como indicamos en la siguiente imagen.



El color de los cables de la cámara podemos apreciarlo en la imagen de Conexión por el puerto RS232 que hay más arriba.

Y el esquema de relación de tiempo de pulso y grados, necesarios para mover el motor lo vemos en la siguiente imagen.

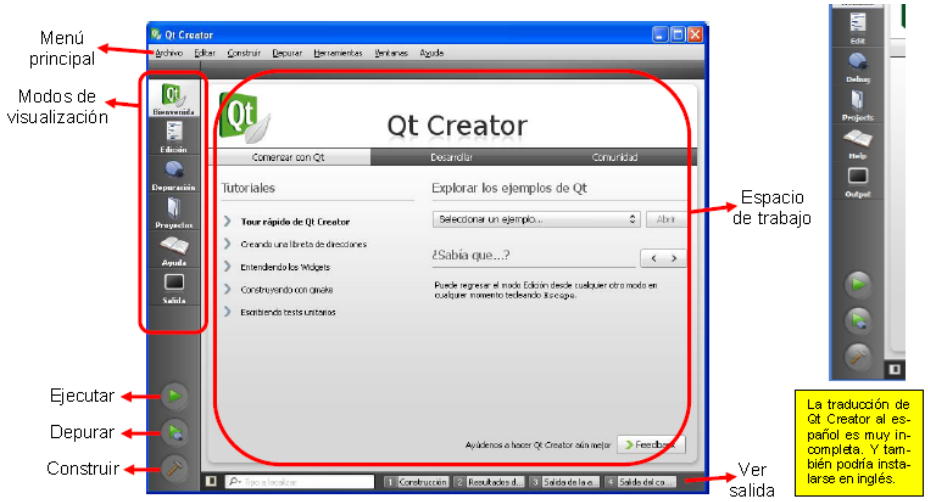


4-Conexión con dispositivo móvil

4.1) El entorno de programación QT

Para realizar la conexión con el dispositivo móvil utilizaremos un router inalámbrico al que conectaremos la cámara, y posteriormente accederemos desde nuestro móvil, esto simplifica el acceso ya que la cámara tiene una ip de la red de la upv y es más sencillo modificar la IP de nuestro router que continuamente la de la cámara ya que es utilizada por mas personal. Para conectar la cámara con el router basta con enchufar la clavija Ethernet de la cámara en una de las conexiones que en el router existen y conectar las alimentaciones, posteriormente procederemos a conectar nuestro dispositivo móvil y PC con el router. Para realizar la aplicación necesaria para que la cámara sea accesible desde nuestro móvil, descargaremos un entorno de desarrollo llamado QT de la página web: ***http://qt.nokia.com/products*** con sus respectivas librerías de: ***http://qt.nokia.com/downloads*****,** la versión utilizada para la realización de la aplicación será la versión QT 4.7.4 de 32 bits. Al instalar la aplicación en nuestro pc junto con sus librerías obtenemos un entrono de trabajo como este:

QT es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con una interfaz gráfica de usuario así como también para el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores. QT utiliza el lenguaje de programación c++. Al ejecutar Qt Creator se abrirá una ventana como la mostrada abajo.



Los elementos principales son:

Modos de visualización. Existen seis modos de visualización diferentes, que nos permiten ver la información más adecuada en cada momento, según estemos editando, depurando, ejecutando, buscando ayuda, etc.

Modo Bienvenida (Welcome). Aparece siempre al empezar. Contiene tutoriales de Qt y nos permite abrir proyectos rápidamente.

Modo Edición (Edit). Sirve para editar el código fuente de nuestra aplicación, así como las ventanas y formularios del programa. Es el modo más habitual de visualización cuando estemos escribiendo el programa.

Modo Depuración (Debug). Lo usamos cuando estemos depurando la aplicación. Muestra el código y la información de depuración.

Modo Proyectos (Projects). Permite ver y configurar las opciones del proyecto. Normalmente no necesitaremos tocarlo mucho.

Modo Ayuda (Help). Muestra la ayuda de Qt. Desafortunadamente, no incluye ayuda de C/C++ ni de las STL.

Modo Salida (Output). Sirve para ver la salida del compilador, la salida del programa o el resultado de las búsquedas.

Método de trabajo. El proceso normal de trabajo empezará con la creación de un nuevo proyecto. Después diseñaremos el aspecto gráfico de la ventana (o ventanas) de nuestra aplicación. Escribimos el código usando el modo Edición. Después depuramos y ejecutamos, hasta estar seguros del funcionamiento correcto del programa.

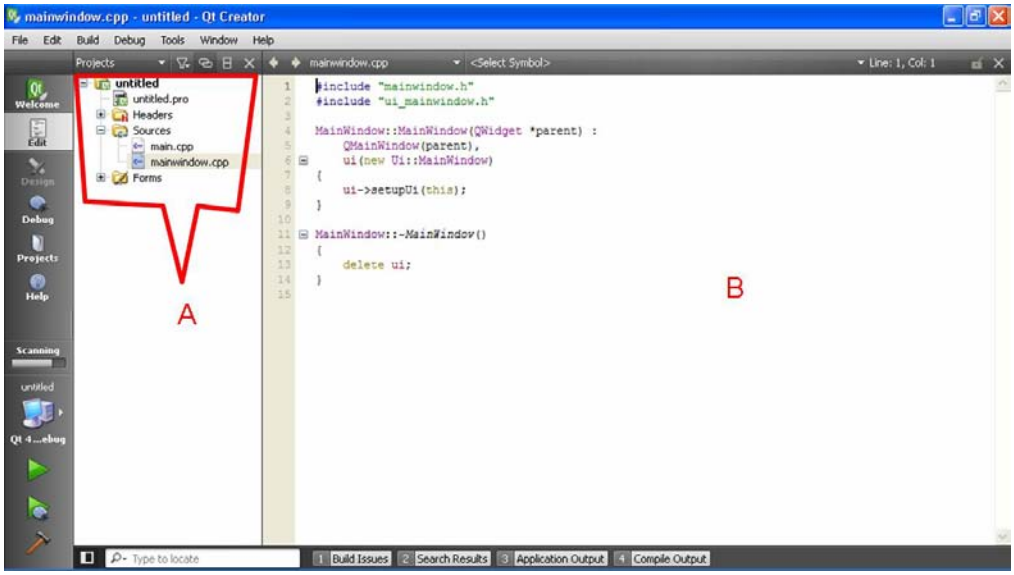
Para realizar un simple ejemplo como un contador con dos botones uno para cuenta ascendente y otro descendente indicando los pasos a seguir crearemos un proyecto como sigue:

1) Crearemos un directorio donde guardar el proyecto en “C:”

2) Abriremos el programa Qt creando un nuevo proyecto seleccionando File->new file or Project; se abrirá un cuadro de dialogo donde seleccionaremos: Qt Widget Project y en la parte derecha mobile Qt aplication, ya que queremos una aplicación para móvil, aceptamos.

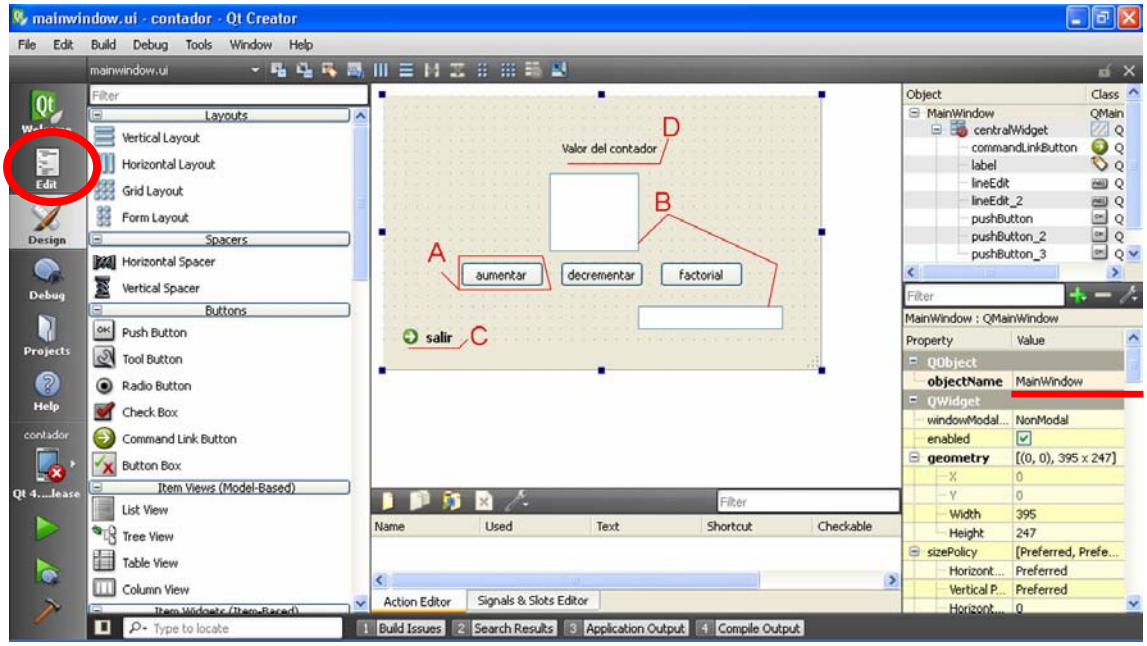
3) En el siguiente cuadro de dialogo, escribiremos el nombre del proyecto y el directorio donde queremos guardarlo que será el creado anteriormente, pinchamos siguiente y elegimos las modalidades de proyecto, para escritorio, móvil o simulador, serán las formas donde podremos simular nuestro programa, pinchamos siguiente, siguiente y finalizar.

4) Ya podemos empezar a desarrollar;

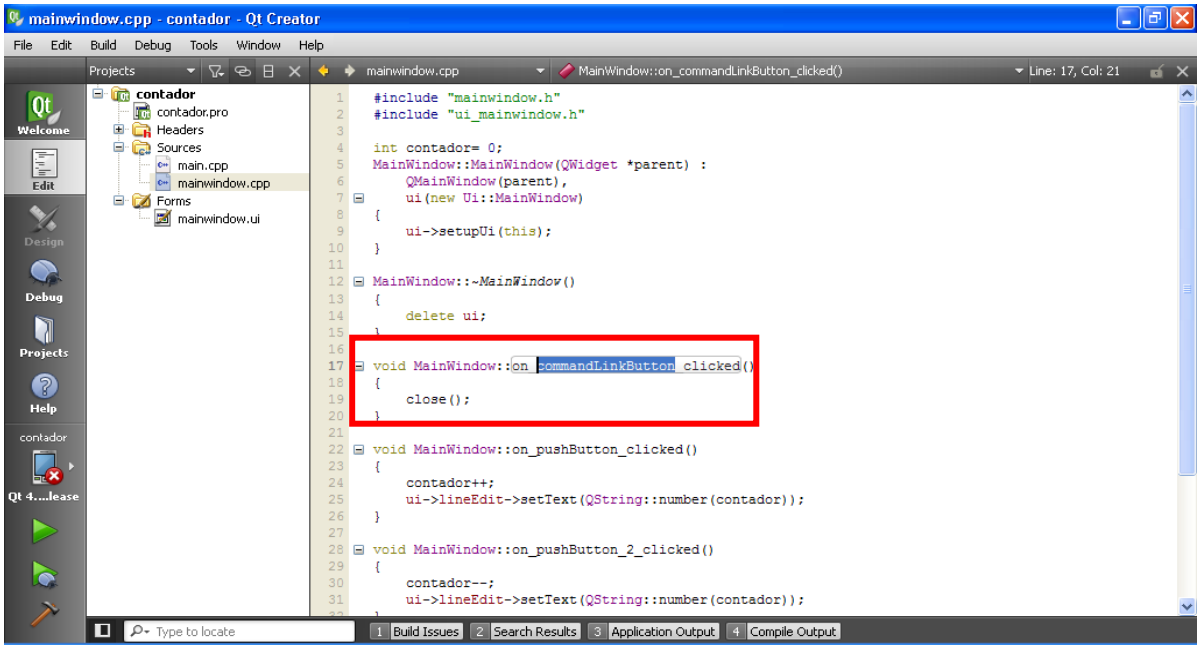


En la parte A tendremos cada uno de los archivos del proyecto interface grafica podemos acceder pinchando sobre Forms código fuente Source y definición de variables en Headers en la parte derecha B aparece el área de trabajo donde montaremos nuestro entorno grafico y completaremos nuestro código.

Empezaremos creando el entorno grafico, un botón para incrementar en 1 y otro para restar a la cifra indicada. También añadiremos un botón para el cálculo del factorial junto con dos recuadros para introducir el número y visualizar el resultado, el botón de salir será para cerrar el programa. En la imagen vienen definidos cada uno de ellos con letras, los botones letra “A”, los recuadros letra “B”, el botón de salir del programa letra “C” y con la letra “D” etiquetas de texto. Todos los elementos se seleccionan del menú de la izquierda y se arrastran hasta el lugar donde queremos ubicarlos.



Haciendo clic con el botón derecho sobre cada elemento en nuestro diseño podemos seleccionar Go To Slot eso agregara un trozo de código a nuestro programa que se encuentra en Sources y que podremos volver a ver pinchando en el icono Edit señalado con un círculo. En la parte derecha recuadrado, aparece el nombre de cada uno de los elementos, así es como se llamarán en el fragmento de código correspondiente. Por ejemplo el botón donde pone “salir” será: “commandLinkButton” en la siguiente imagen vemos el lugar donde aparece el código con la acción que realiza.



El fragmento de código sería;

```
void MainWindow::on_commandLinkButton_clicked()
{
   close();
}
```

Ya tendríamos el código para cerrar la aplicación.

Para realizar una cuenta en el apartado de código correspondiente al botón de cuenta ascendente deberemos introducir:

```
void MainWindow::on_pushButton_clicked()
{
    contador++;
    ui->lineEdit->setText(QString::number(contador));
}
```

y en el de la cuenta descendente:

```
void MainWindow::on_pushButton_2_clicked()
{
    contador--;
    ui->lineEdit->setText(QString::number(contador));
}
```

Si queremos el factorial:

```
void MainWindow::on_pushButton_3_clicked()
{
    long resultado= 1;
    for (int i= 1; i<=contador; i++)
    resultado*= i;
    ui->lineEdit_2->setText(QString::number(resultado));
}
```

Esos fragmentos de código corresponden con botones en nuestro último caso del factorial al realizar un click hará:

```
long resultado= 1;
    for (int i= 1; i<=contador; i++)
    resultado*= i;
```

mostrando el resultado por `ui->lineEdit_2` (recuadro de texto de la interface grafica)

Una vez realizado nuestro proyecto y antes de probarlo en nuestro PC o móvil deberemos realizar una serie de configuraciones.

Pincharemos en el apartado *projects* de la parte izquierda apareciendo una pantalla de esta forma:

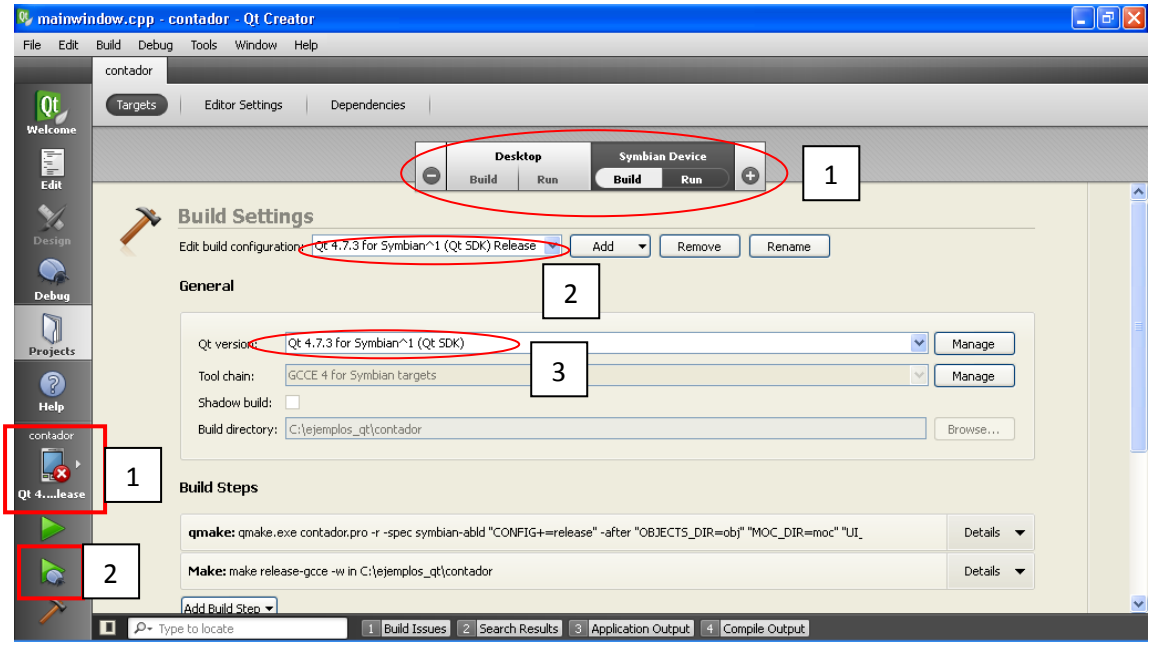


Imagen entorno

En el número 1 podemos encontrar las distintas formas de simulación: desktop (escritorio del pc), symbian Device(movil) y Qt simulator(simulación desde el pc pero con interface del móvil).

En el numero 2 seleccionaremos *Qt 4.7.3 for symbian^1(Qt SDK).release* o *debug*, una de las dos.

En el numero 3 encontraremos la versión de simulador en nuestro caso para symbian device utilizaremos la *Qt 4.7.3 for symbian^1(Qt SDK).*

Una vez configurado nuestro programa, configuraremos nuestro dispositivo móvil para poder ejecutar aplicaciones realizadas por Qt, nuestro dispositivo móvil debe tener como SO el sistema Symbian propio de los móviles Nokia, para poder instalar en el móvil los drivers necesarios tendremos que tener instalado en el pc el software Nokia Pc suite el cual nos permite comunicar el pc con el dispositivo para la trasferencia de archivos. Conectaremos el móvil con el cable apropiado a nuestro pc, en el menú de inicio pinchamos sobre *Qt sdk->symbian^1->Install Qt 4.7.3 for Symbian^1 on device* y

Qt sdk->symbian^1-> Install QtWebKit for Qt 4.7.3 on Symbian^1 device , con esto será suficiente para poder ejecutar aplicaciones en nuestro móvil.

La simulación del programa contador utilizando la opción Qt simulator se realiza seleccionando en el apartado recuadrado en rojo numero 1 en la imagen superior *"entorno"* la opción correspondiente a *QT simulator (release o debuj),* una vez seleccionada y pulsando sobre el icono recuadro 2 el programa se compila y se ejecuta apareciendo los elementos que en la siguiente imagen se indican:

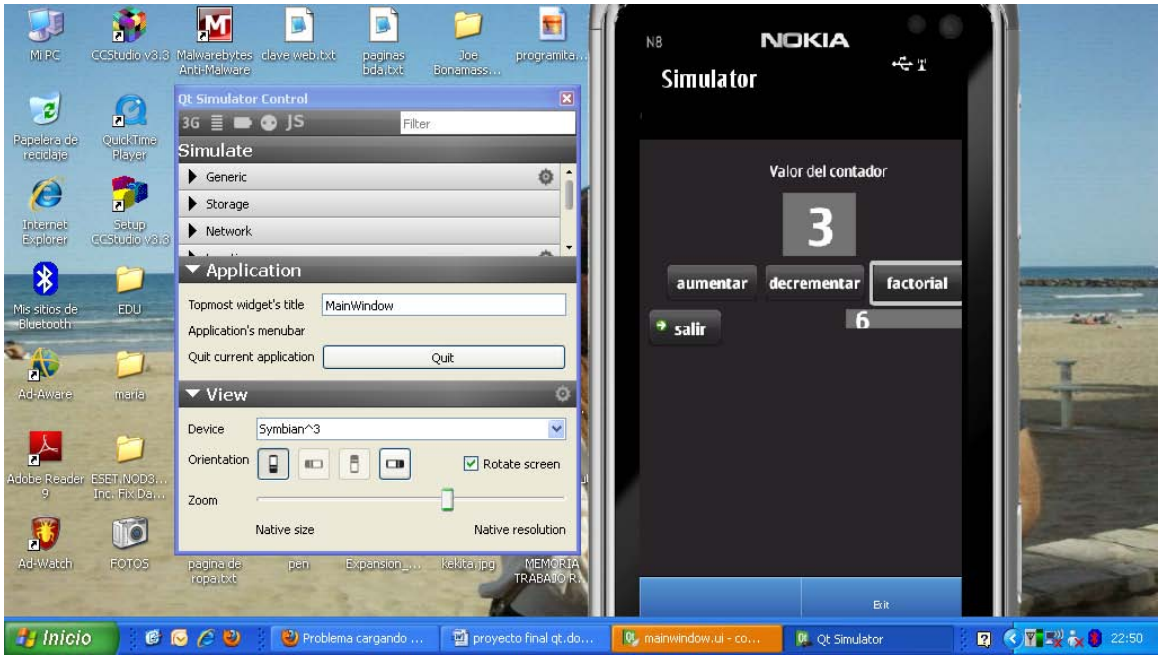


Imagen Qt Simulator

4.2) Programa de conexión con smart phone Nokia

Para la realización de nuestro proyecto tendremos que seguir los mismos pasos pero añadiendo mas pantallas graficas y mas fuentes en el apartado Sources con sus respectivas headers. El programa se llamara TeraTermobile.

Constara en la carpeta Sources de:

cliente.cpp ->lugar donde van declaradas todas funciones respecto a establecer la comunicación mediante socket y efectuar envíos y recepciones de datos.

Ejemplo:

```
void cliente::conectarse(QString host, int port)
{
    this->socket->abort();
  this->socket->connectToHost(host, port);

void cliente::sendMessage(QString msg)
{
    //READ: Si el socket de la conexion es valido entonces enviar el
mensaje
    // Pero convertirlo en ASCII
    if (socket->isValid())
    socket->write(msg.toAscii());
```

También incluimos la función que se encarga de formar la imagen para visualizarla en el móvil, debido a que es más simple ya que se utilizan funciones de captura de datos que se encuentran en dicho cpp. En dicho fragmento de código podemos destacar el doble bucle for que ordena los pixeles.

```
   for(int i =0; i <image2->height(); ++i){
             row=image2->scanLine(i);
             for(int j =0; j < image2->width(); ++j) {
                 row[j] = array[i*image2->width() + j] ;
               }
             }
```

comandos.cpp -> corresponde con la pantalla donde se envían los comandos a la cámara, se encuentran las funciones de botones que aparecen en dicha pantalla.

Ejemplo:

```
void comandos::on_lineEdit_2_returnPressed()
{
    if(this->Conexion->estaConectado)
    {
        if(this->ui->lineEdit_2->text().trimmed()!="")
            this->Conexion->sendMessage(this->ui->lineEdit_2->text() +
" \n\r");
    }this->ui->lineEdit_2->clear();
}
```

Imagen.cpp -> corresponde al código de botones de la pantalla donde se visualiza la imagen formada que envía la cámara.

Ejemplo:

```
void imagen::on_pushButton_clicked()
{
   this->Conex->sendcabecera( this->ui->lineEdit_2->text(),
                              this->ui->lineEdit->text().toInt());}
```

Este fragmento corresponde con un checkBox el cual al estar marcado guarda la imagen en el directorio que indica.

```
void imagen::handleReceivedImage(QPixmap& image)
{
    ui->receivedImage->setPixmap(image);
    if(save==true){image.save("E:/fot.jpg");}
}
```

load.cpp -> corresponde con la pantalla donde se selecciona el archivo a enviar a la cámara.

Ejemplo:

```
void load::on_pushButton_2_clicked()
{
    if (ui->tableWidget->rowCount() == 0)
        return;
    if (ui->tableWidget->rowCount()==1 )
    {
        ui->lblTamanio->setText("0");
    }
    else
    {
        QTableWidgetItem *item;
        item=ui->tableWidget->
                item(ui->tableWidget->currentRow(),2);
        tamanio=tamanio - item->text().toFloat();
        ui->lblTamanio->setText(QString::number(tamanio));
    }
    ui->tableWidget->removeRow(ui->tableWidget->currentRow());
}
```

MainWindow.cpp -> es una pantalla inicial con un solo botón que te introduce al programa.

Ejemplo:

```
void MainWindow::on_commandLinkButton_clicked()
{
    close();
}
void MainWindow::on_pushButton_2_clicked()
{
    comandos c;
    c.exec();
}
```

Algunas de las fotos del programa en funcionamiento:

Objeto del que captaremos la imagen y vista de la posición de la cámara VC4018

Pantalla donde se establece la conexión con la cámara, escribiendo IP y puerto, línea para introducir comandos y espacio donde podemos ver la respuesta en texto de la cámara, y botón para cargar archivos.

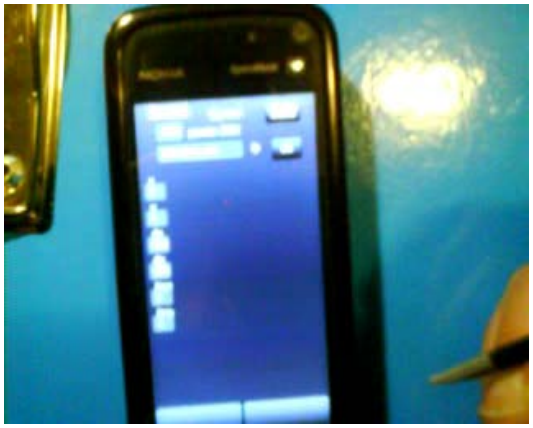
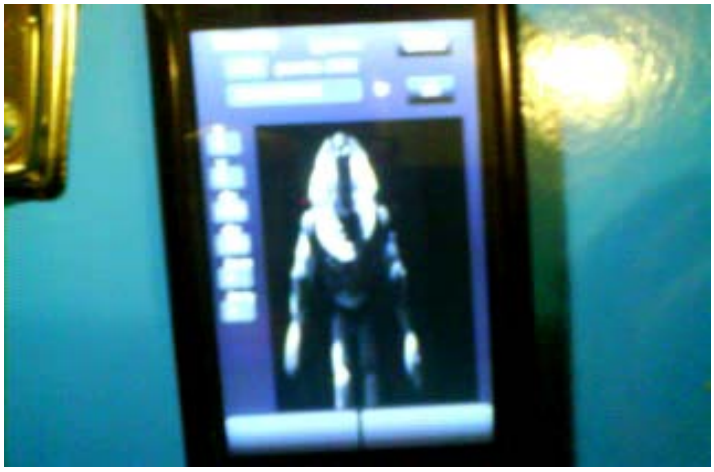


Imagen A

Imagen B

Accedemos mediante un botón a la pantalla imagen, (imagen B) después de enviar el respectivo comando, en esta pantalla visualizaremos la respuesta de la cámara en imágenes, permitiendo almacenarlas en la memoria del teléfono simplemente marcando la casilla save.



Resultado de procesar la imagen enviada por la cámara al dispositivo, al pulsar exit se corta la comunicación con la cámara.

5-Conclusiones y trabajos futuros.

Con este proyecto hemos podido aplicar conocimientos adquiridos durante la carrera en algunas de sus asignaturas, en materia de desarrollo de aplicaciones como son los conocimientos adquiridos en la asignatura de Ingeniería de Software, en la asignatura de Redes para la comunicación entre dispositivos mediante sockets y funcionamiento de protocolos de comunicación, conocimiento de transmisión de datos aplicado en la asignatura de Sistemas de Entrada Salida o la formación de imágenes digitales en la asignatura de Sistemas de Visión por Computador y muchas otras más que aportan pequeños detalles que hacen comprender mejor el funcionamiento de las herramientas utilizadas.

Hemos puesto en marcha un sistema de tecnología cada vez más demandada en infinidad de aplicaciones y tareas, ya sea en control de procesos industriales, como vigilancia, o como sustitución de las personas en tareas peligrosas, etc.

Usando un pequeño procesador incluido en una cámara que potencia su funcionalidad y un dispositivo móvil que te permite actuar a distancia operando a través de la red.

Esta aplicación puede desarrollarse todavía más ampliando su utilidad y llegando a realizar aplicaciones totalmente asombrosas, como podía ser conducir un vehículo a distancia controlado por un móvil y con visión basada en una cámara como la del proyecto presente.

Los principales pasos que se han seguido para el desarrollo del proyecto han sido:

• Aprender el funcionamiento de los distintos equipos que se han utilizado para el diseño del sistema.
• Montaje e implementación del sistema de captura
• Realizar la Programación y manipulación de programas en **C** para la cámara.
• Aplicar conocimientos adquiridos en la carrera realizando una aplicación mediante Qt de comunicación entre dispositivos conectados a la red.

6-Apéndices

• Características de la cámara a color.
También he realizado una búsqueda por categorías en la web de vision-components, obteniendo una lista de la base de datos de todos los artículos relacionados tanto de software como de hardware de las cámaras que disponemos. Para ello hay que estar registrado

Ref: http://vision-components.com/nc/de/service-support/wissensdatenbankfaq/faq-vc20xxvcsbc40xxvc4xxx/?sword_list[0]=vc&sword_list[1]=cameras&sword_list[2]=with&sword_list[3]=color&sword_list[4]=sensor

Mirando la lista encontramos, un artículo. El artículo se llama: VC Snart Cameras with color sensor relacionado con las cámaras con sensor de color.
Este artículo explica las diferentes cámaras a color de visión -components, además de cómo realizar el balance de blancos en una cámara a color; viene con links de descarga de los programas para poder ejecutar en la cámara.
Señalar que también hay otros artículos de interés.

• Balance de blancos.
Vision-Components vende dos tipos de cámaras a color:
Cámaras de color reales: por ejemplo VC2065EC, y la VC4065 / VC4465EC.
Cámaras con sensor de color: por ejemplo VC4016C, VC4018C, SBC4016C, SBC4018C y, teóricamente, todos los modelos cámaras VC4xxx pedidas con sensor de color. El modelo utilizado en el laboratorio viene provista de filtros bayer.
El programa **“wb.c”** sirve para ajustar el balance de blancos para cámaras de color.
Y por otro lado está el "whitebal.c" para ajustar el balance de blancos en las cámaras con sensores de color (especialmente VC4016/4018E).
Pero este programa no podemos compilarlo con el C.Cv3.3 porque nos faltan librerías.

No reconoce: La function “WhiteBalanceValues”, “init_color_lut”.
Se puede ver su estructura en la siguiente referencia:
Ref: color_lib.pdf (pág. 22 y 23)
La conclusión es, que a aparte de tener las bibliotecas instaladas VCLIB.LIB, VCLIB.H, FLIB.LIB y FLIB.H hay que instalar el paquete de librerías VCLIB para color. También es necesario una licencia válida para VCLIB y es necesario llamar a las funciones init_licence () y init_vclib () en este orden en los programas.

La bibliografía utilizada para la realización de este proyecto, se ha basado la mayoría en recursos de Internet y en manuales de la cámara. Documentación interactiva encontrada en la Red. Además de la información aportada por Ángel Rodas Jordá.