VISIÓN ARTIFICIAL

LABORATORIO SESIÓN 2

Captura de vídeo e imágenes

LABORATORIO SESIÓN 2.ÍNDICE

- 2.1 Cámara
- 2.2 Óptica
- 2.3 Captura de imagen y vídeo
- 2.4 Otras consideraciones
- 2.5 Objetivo práctico

LABORATORIO SESIÓN 2.ÍNDICE

- **2.1 Cámara**
- 2.2 Óptica
- 2.3 Captura de imagen y vídeo
- 2.4 Otras consideraciones
- 2.5 Objetivo práctico

2.1 Cámaras

- Modelo Firefly MV de PointGrey (color): **FMVU-03MT**

Cámara

Tapa

Adaptador trípode

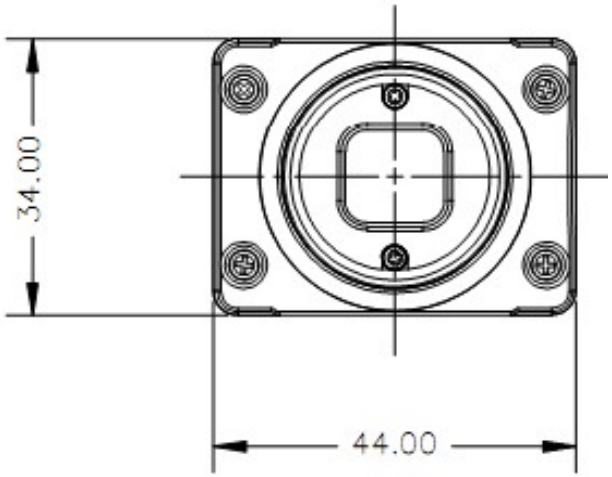
USB 2.0

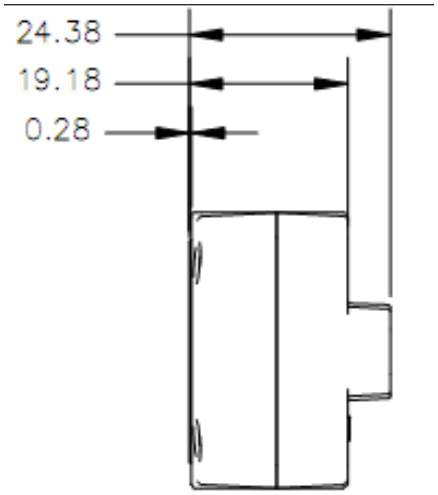
INTERFAZ CONTROL

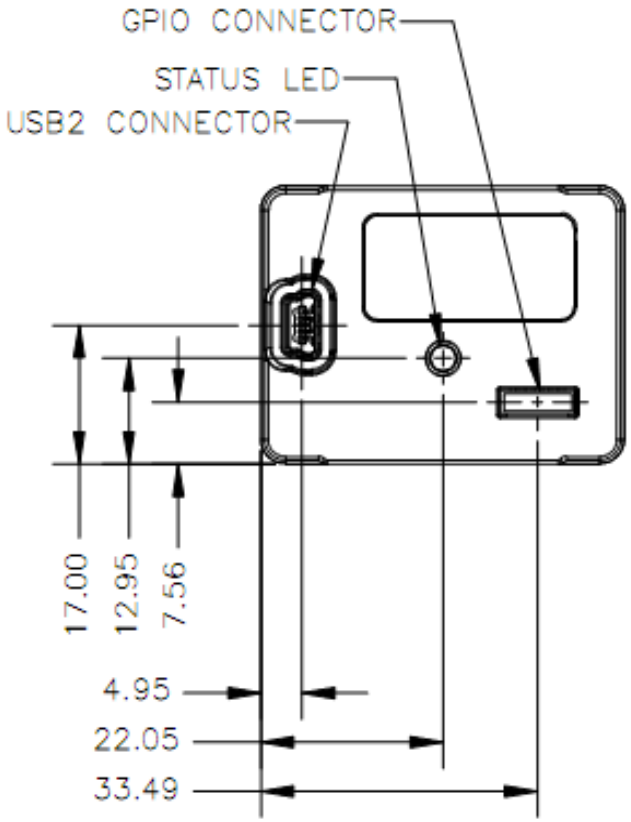
Adaptador 5mm para montura tipo C

2.1 Cámaras

- Dimensiones físicas







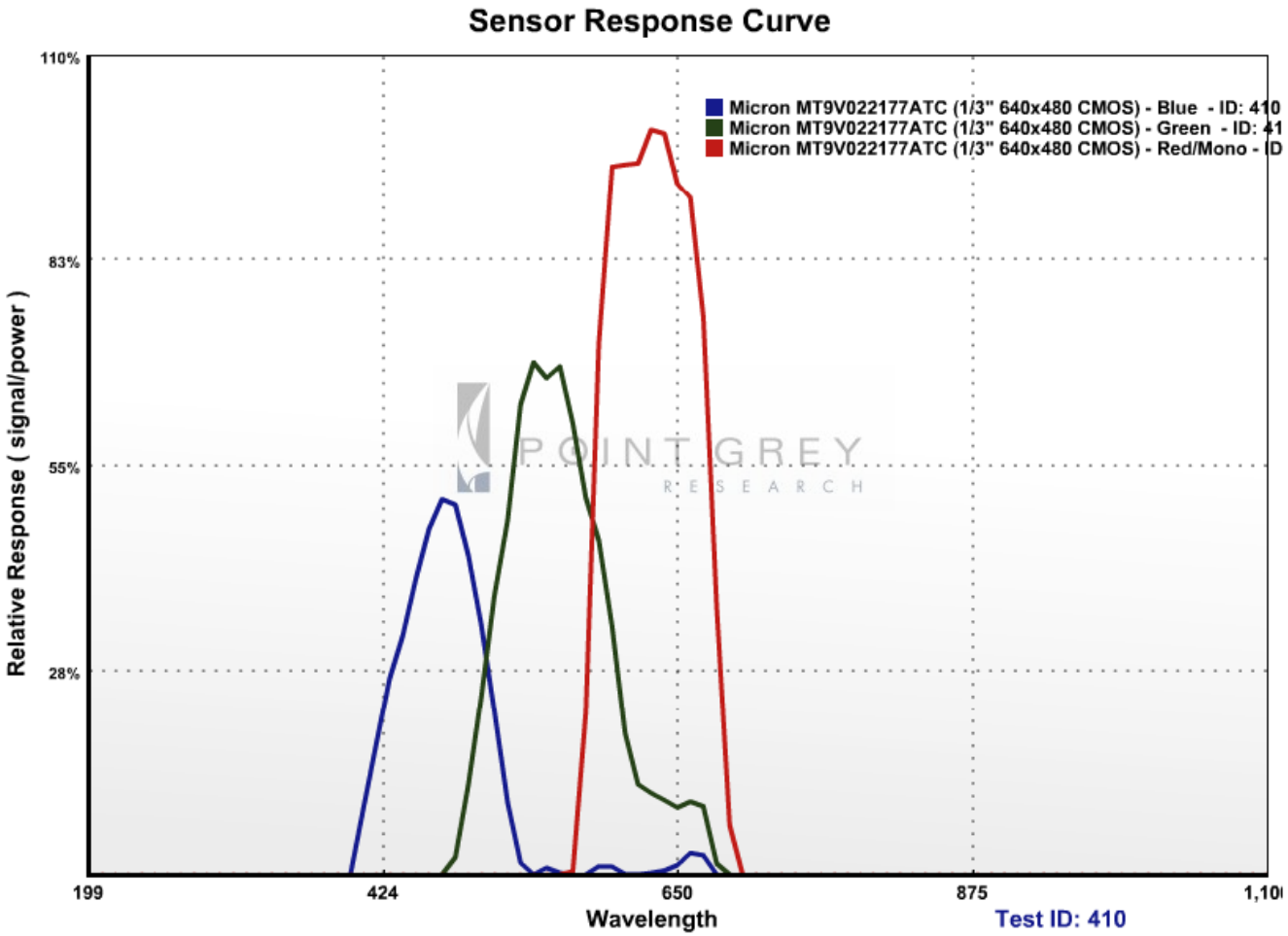
2.1 Cámaras

- Características

Imaging Sensor	Micron 1/3" Wide-VGA CMOS
	MT9V022177ATM (BW) MT9V022177ATC (Color)
Shutter Type	Global shutter using Micron TrueSNAP™ technology
Active Imager Size	4.51mm (H) x 2.88m (V), Diagonal 5.35mm (1/3" type)
Active Pixels	752(H) x 480(V)
Pixel Size	6µm(H) x 6µm(V)
A/D Converter	On-chip 10-bit analog-to-digital converter
Video Data Output	8 and 16-bit digital data (see *Supported Data Formats* below)
Standard Resolutions	640x480
Frame Rates[1]	60, 30, 15, 7.5 FPS
Partial Image Modes	Pixel binning and region of interest modes available via Format_7
Interfaces	5-pin Mini-B USB 2.0 for camera control, video data transmission and power
	7-pin JST GPIO connector, 4 pins for trigger and strobe, 1 pin +3.3 V, 1 V_{EXT} pin for external power
Voltage Requirements	4.75 to 5.25 V via the Mini-B USB 2.0 cable or JST 7-pin GPIO connector
Power Consumption	Less than 1W
Gain	Automatic/Manual Gain modes
	0dB to 12dB
Shutter	Automatic/Manual Shutter modes
	0.03 ms to 512 ms (extended shutter mode)
Gamma	0 to 1 (enables 12-bit to 10-bit companding)
Trigger Modes	IIDC v1.31 Trigger Modes 0 and 3
Signal To Noise Ratio	52 dB
Dimensions	44 mm x 34 mm x 24.38 mm (case enclosed)
Mass	37 grams (including tripod adapter)
Camera Specification	IIDC 1394-based Digital Camera Specification v1.31
Emissions Compliance	Complies with CE rules and Part 15 Class B of FCC Rules.
Operating Temperature	Commercial grade electronics rated from 0° - 45°C
Storage Temperature	-30° - 60°C
Operating Relative Humidity	20 to 80% (no condensation)
Storage Relative Humidity	20 to 95% (no condensation)

2.1 Cámaras

- Curva de respuesta



2.1 Cámaras

- Propiedades generales de la cámara

Property	Min	Max	Auto	On/Off	One Push	Absolute Mode	Defaults
Brightness	1	255	Y	N	N	N	Auto, On
Exposure	7	62	Y	Y	N	N	Auto, On
Gamma	0	1	N	Y	N	N	Off
Pan	0	112	Y	Y	N	N	Auto, On
Shutter	0.06 ms	33.19 ms	Y	N	N	Y	Auto, On
Gain	0 dB	12.04 dB	Y	N	N	Y	Auto, On
White Balance (COL)	1	1023	N	Y	N	N	On
Frame Rate	4.6 FPS	30.42 FPS	Y	Y	N	Y	Auto, On, 30 FPS

2.1 Cámaras

- Flujo de datos

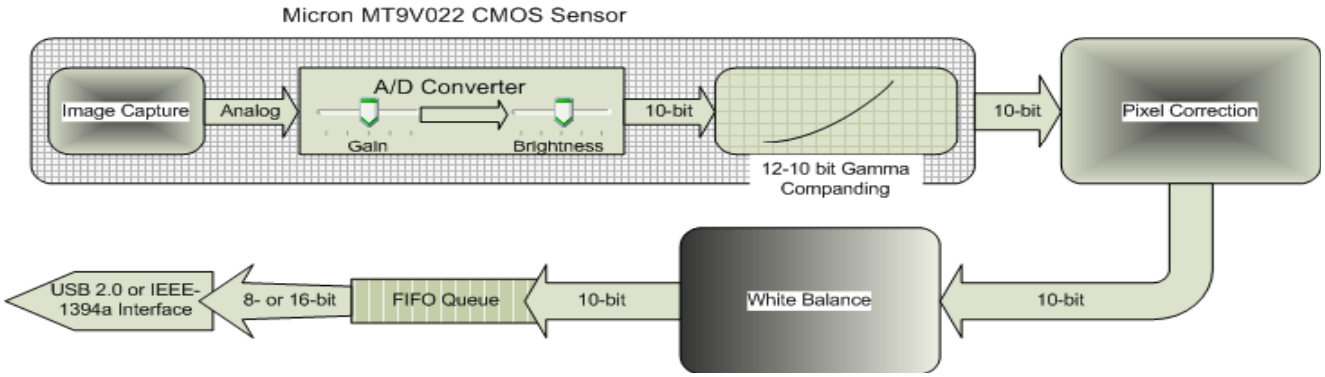


Image Data Flow Step	Description
Sensor	Image capture, analog-to-digital conversion and gamma adjustment (FFMV-03M2/FMVU-03MT only) all take place on board the camera sensor.
Analog to Digital (A/D) Converter	The sensor's A/D Converter transforms pixel voltage into a 10- or 12-bit value, adjusting for gain and brightness in the process. Gain and brightness cannot be turned off.
Gamma	The *Firefly MV* supports gamma adjustment to reduce noise at low light levels. For more information, see Section 3.6.2 Lookup Table and Gamma. The gamma setting of the camera's default memory channel is OFF, and no correction occurs.
Pixel Correction	The camera firmware corrects any blemish pixels identified during manufacturing quality assurance by applying the average value of neighboring pixels. For more information, see Knowledge Base Article 314.
White Balance	In color models, color intensities can be adjusted manually to achieve more correct balance. The white balance setting of the camera's default memory channel is ON. If not ON, no white balance correction occurs.
FIFO Queue	The final output of image data is controlled in a first-in, first-out (FIFO) queue.
1394a or USB 2.0 Interface	Depending on your camera's interface, data is transferred at the following rates: • 480 Mbit/s via a 5-pin Mini-B USB 2.0 port • 400 Mbit/s via a 6-pin IEEE-1394a port

LABORATORIO SESIÓN 2.ÍNDICE

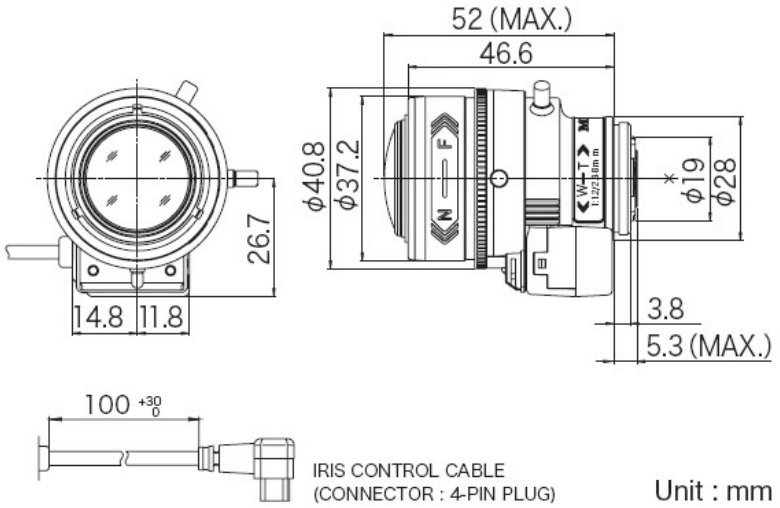
- 2.1 Cámara
- **2.2 Óptica**
- 2.3 Captura de imagen y vídeo
- 2.4 Otras consideraciones
- 2.5 Objetivo práctico

2.2 Óptica

YV2.8x2.8SA-SA2

2.8 x

Applicable camera (model): 1 | 2/3 | 1/2 | 1/3 | 1/4



Vari-Focal | Wide Angle | DC Auto Iris | CS Mount | Metal Mount | ND Filter | Long Cable | Aspherical Lens | Large Aperture Ratio | RoHS Compliant

VARI | WIDE | DC | CS-mt | METAL | ND | 230 | AT | F1.2 | RoHS

HD 3MP

2.2 Óptica

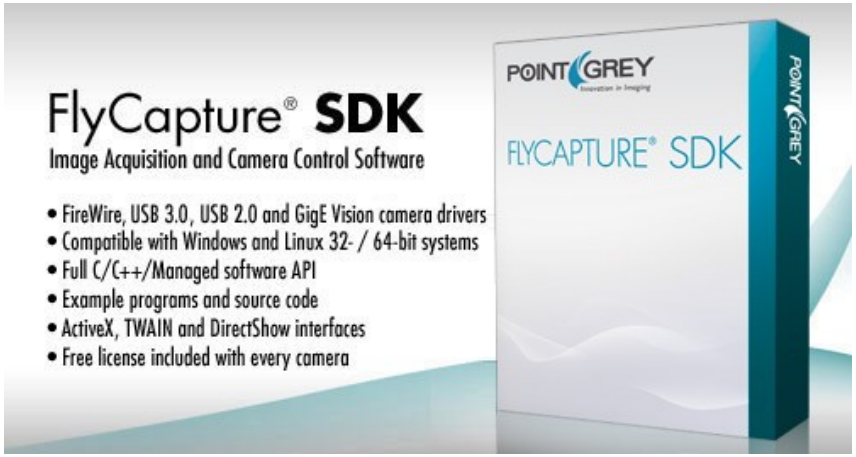
Focal Length (mm)			2.8~8 (2.8×)
Iris Range			F1.2 ~ Close
Operation	Zoom		Manual
	Focus		Manual
	Iris		Manual
Angle Of View (H×V)	1/3"	WIDE	100°00′ × 73°45′
		TELE	35°03′ × 26°18′
	1/4"	WIDE	73°45′ × 54°49′
		TELE	26°18′ × 19°44′
Angle Of View (H×V) Aspect Ratio 16:9	1/3"	WIDE	109°50′ × 59°51′
		TELE	38°11′ × 21°29′
	1/4"	WIDE	80°39′ × 44°38′
		TELE	28°39′ × 16°07′
Focusing Range (From Front Of The Lens) (m)			∞ ~ 0.3
Object Dimensions at M.O.D. (H×V) (mm)	1/3"	WIDE	744 × 468
		TELE	197 × 146
	1/4"	WIDE	468 × 323
		TELE	146 × 108
Object Dimensions at M.O.D. (H×V) (mm) Aspect Ratio 16:9	1/3"	WIDE	890 × 359
		TELE	216 × 118
	1/4"	WIDE	530 × 256
		TELE	159 × 88
Back Focal Distance (in air) (mm)			7.70
Exit Pupil Position (From Image Plane) (mm)			314
Filter Thread (mm)			—

LABORATORIO SESIÓN 2.ÍNDICE

- 2.1 Cámara
- 2.2 Óptica
- **2.3 Captura de imagen y vídeo**
- 2.4 Otras consideraciones
- 2.5 Objetivo práctico

2.3 Captura de imagen y vídeo

- Software Development Kit (SDK) proporcionado por PointGrey
  - Librerías en C/C++ para Windows y Linux
  - Compatible con otras librerías abiertas tipo IIDC 1394
  - Optimizado para las cámaras de PointGrey



2.3 Captura de imagen y vídeo

- Obtener imágenes de la cámara

```
FlyCapture2::Camera;
Error Connect( PGRGuid* pGuid = NULL );
Error StartCapture( ImageEventCallback callbackFn = NULL, const void* pCallbackData = NULL );
Error RetrieveBuffer( Image* pImage );
Error StopCapture();
Error Disconnect();
```

```
FlyCapture2::Image;
Error Convert( PixelFormat format, Image* pDestImage );
unsigned int GetRows();
unsigned int GetCols();
unsigned char* GetData();
```

```
FlyCapture2::Camera cam;

// connect + startcapture
FlyCapture2::Image rawImage, convertedImage;

error = cam→RetrieveBuffer(&rawImage); // get an image from camera


error = rawImage.Convert(FlyCapture2::PIXEL_FORMAT_MONO8, &convertedImage);
cv::Mat imgRead =
cv::Mat(convertedImage.GetRows(),convertedImage.GetCols(),CV_8UC1,convertedImage.GetData());
```

2.3 Captura de imagen y vídeo

- **Cadenas de caracteres con formato**
  - `char nombreImagen[100];`
  - `sprintf(nombreImagen, "./dat/imagen%04d.png", globalIndexImages++);`

- **Guardar imagen**
  - Formatos soportados
    - Windows bitmaps - BMP, DIB
    - JPEG files - JPEG, JPG, JPE
    - Portable Network Graphics - PNG
    - Portable image format - PBM, PGM, PPM
    - Sun rasters - SR, RAS
    - TIFF files - TIFF, TIF

```
cv::imwrite(nombreImagen, imgRead);
```

2.3 Captura de imagen y vídeo

- **Guardar video**
  - Lo guardamos en “crudo” con **fwrite** en ficheros con extensión RAW (.raw)
    - Se abre un vídeo
    - Se guardan los *frames* necesarios de forma lineal en memoria
    - Se cierra el vídeo
  - Para recuperar el vídeo necesitamos saber el **ancho**, el **alto** y el **número de canales**

```
int globalIndexVideo=0; // variable global inicialmente a 0
char nombreVideo[100];

sprintf (nombreVideo, "video%04d.raw", globalIndexVideo++);
pf_video = fopen(nombreVideo, "w"); // solo se hace una vez por video

...
// por cada frame
fwrite(imgRead.data, imgRead.rows*imgRead.cols, sizeof (unsigned char), pf_video);
...


fclose(pf_video); pf_video=NULL; // solo se hace una vez por video
```

2.3 Captura de imagen y vídeo

- Leer imágenes de fichero (*se proporciona*)

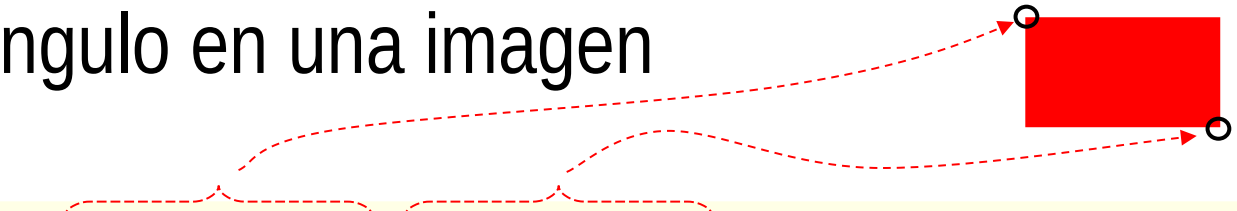
```
int main(int argc, char** argv){
  FILE *pf_video = NULL;
  int numCols=640, numRows=480;
  char q;

  if(argc!=4) {
    printf("\n Introduzca el nombre del video, y el tamaño (ancho y alto)\n");
    printf("\n      Ejemplo: ./lee video.raw 640 480\n");   exit(-1);  }

  pf_video = fopen (argv[1],"r");
  numCols = atoi(argv[2]);
  numRows = atoi(argv[3]);

  cv::Mat imageVideo = cv::Mat(numRows,numCols,CV_8UC1);
  while ((fread(imageVideo.data, sizeof (unsigned char), numRows * numCols, pf_video)) > 0 &&
     key != 'Q') {

     cv::imshow(string(argv[1]), imageVideo);
     char aux = cv::waitKey(33);
     key = toupper(aux);
     switch(key) {
          case 'P': cv::waitKey(0); break;
     }
  }
  fclose(pf_video);
  return 0;
}
```

LABORATORIO SESIÓN 2.ÍNDICE

- 2.1 Cámara
- 2.2 Óptica
- 2.3 Captura de imagen y vídeo
- **2.4 Otras consideraciones**
- 2.5 Objetivo práctico

2.4 Otras consideraciones

- Dibujar rectángulo en una imagen



```
cv::rectangle(imgReadColor,cv::Point(10,10),cv::Point(50,50),CV_RGB(255,0,0),CV_FILLED);
```

- Convertir una imagen en escala de grises a color

```
cv::cvtColor(imgRead,imgReadColor, CV_GRAY2RGB);
```

- Escribir un texto sobre una imagen

```
cv::putText(imgReadColor, //imagen sobre la que se desea escribir el texto
          “REC”,  //texto que se desea escribir
          cvPoint(60, 50),  //posición del texto
          cv::FONT_HERSHEY_COMPLEX_SMALL, //tipo de fuente
          2.5, //tamaño del texto
          CV_RGB(255,0,0), //color del texto
          3); //grosor de la linea del texto
```

LABORATORIO SESIÓN 2.ÍNDICE

- 2.1 Cámara
- 2.2 Óptica
- 2.3 Captura de imagen y vídeo
- 2.4 Otras consideraciones
- **2.5 Objetivo práctico**

2.5 Objetivo práctico

- Descargar `Sesion2.zip` y descomprimirlo en:
  - `/home/l8/`
- El fichero contiene 2 carpetas:
  - **LeeVideo**
    - Este programa consta de la función `main` descrita en la trasparencia 18 en un solo fichero `main.cpp` (se proporciona un `Makefile` también)
  - **CapturaVideo.zip**
    - `main.cpp`
    - `CaptureCode.cpp, CaptureCode.h`

2.5 Objetivo práctico

- `main.cpp`
  - Inicializa el Bus y la cámara
  - Llama a runSingleCamera()
- `CaptureCode.cpp, .h`
  - Se conecta con la cámara
  - Inicializa la captura
  - Entra en un bucle `while`
    - Obtiene imagen
    - La convierte a `cv::Mat`
    - La muestra con `cv::imshow`

2.5 Objetivo práctico

- **Se pide**: modificar **CaptureCode.cpp** (donde pone **TO DO**…) para que el programa haga lo siguiente:
  - Si se pulsa la tecla ‘s’ (o ‘S’) que guarde una imagen formato PNG
  - Si se pulsa la tecla ‘g’ (o ‘G’) que grabe un vídeo en formato RAW y que pare cuando se vuelva a pulsar la tecla ‘g’ (o ‘G’).
    - Cuando esté grabando tiene que aparecer un rectángulo rojo de 40x40 píxeles en la posición (10,10) de la imagen, y al lado la palabra REC

    - **Idea**: montar una máquina de estados con 2 estados (grabando/sin grabar) gobernados por la tecla ‘g’ (o ‘G’)
  - Comprobar su funcionamiento usando el programa **Lee**