Practica 3:

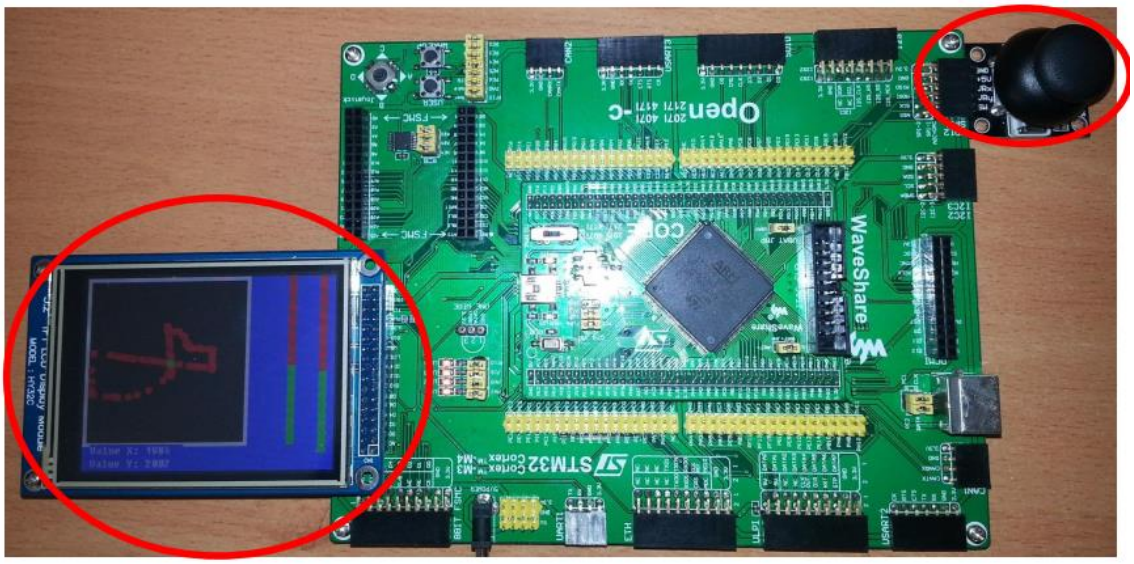
Displays LCD gráficos para sistemas empotrados

Alejandro Vega
Antonio Portillo

Objetivos

- Controlar un display LCD usando el STM32:
  - Texto.
  - Gráficos estáticos.
  - Gráficos dinámicos.
- Diseño e implementación de tareas que muestren distintos gráficos interactivos.
- Creación y eliminación de tareas dinámicamente.
- Lectura del ADC mediante DMA.

Hardware Setup



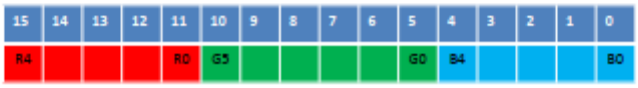
Desarrollo de práctica

1. Manejo del display

- **Tamaño:** 3.2 pulgadas
- **Resolución espacial:** 320 horizontal X 240 Vertical
- **Resolución color:** 16bits (RGB565)
- **Touch Panel:** Resistivo, con interfaz SPI (ADS7843)

La codificación RGB-565

- El display tiene 16 bits de resolución de color y usa la codificación RGB-565.
- Los 16 bits del color empaquetan el color usando la codificación 565:
  - Los 5 bits más significativos contienen el rojo (32 niveles de rojo).
  - Los 6 bits siguientes contienen el verde (64 niveles de verde).
  - Los 5 bits menos significativos contienen el azul (32 niveles de azul).



Funciones

uint16_t color = RGB565CONVERT(R,G,B);

LCD_Initialization(); //Inicializa el FSMC y el display

LCD_Clear (Blue); //Limpia la pantalla en azul

LCD_SetPoint (x, y, color); //Rellena un pixel

LCD_PutChar (x, y, 'b', color, colorFondo); // Mostrar un carácter suelto

LCD_PrintText(x,y,"cadena",color, colorFondo); //Mostrar una cadena de texto

LCD_DrawLine(x1,y1,x2,y2,color) // Pintar una línea

LCD_DrawRectangle(x,y,ancho,alto,pxborde,color) // Pintar un rectángulo

LCD_DrawCircle(x,y,radio,color) *// Pintar un círculo completo con centro en (x, y)*

LCD_FillRectangle(x,y,ancho,alto,color) *// Rellenar una rectángulo*

LCD_FillCircle (x, y, radio, color) *// Rellenar un círculo competo con centro en (x, y):*

2. Ejercicio 1

Modificar **LCDHelloWorldTask** para pintar los siguientes elementos antes de entrar en el bucle:

- Rellenar rectángulo con origen 100, 100, y un tamaño de 100x20 de color amarillo.
- Rellenar rectángulo con origen 150, 80, y un tamaño de 30x20 de color amarillo.
- Dibujar un rectángulo simple en 155, 85 de 20x10, con 2 píxeles de borde, y color rojo.
- Rellenar círculos:
  - En 120, 120 con radio 10 de color negro.
  - En 180, 120 con radio 10 de color negro.
  - En 120, 120 con radio 5 de color blanco.
  - En 180, 120 con radio 5 de color blanco.

```
char str[32];
uint16_t i=0;
uint16_t a=0;
LCD_Clear(Red);
LCD_PrintText(10,20, "Hola Mundo!", Blue, White);

LCD_FillRectangle( 100, 100, 100, 20, Yellow);
LCD_FillRectangle( 150, 80, 30, 20, Yellow);
LCD_DrawRectangle( 155, 85, 20, 10, 2, Red);
LCD_FillCircle( 120, 120, 10, Black);
LCD_FillCircle( 180, 120, 10, Black);
LCD_FillCircle( 120, 120, 5, White);
LCD_FillCircle( 180, 120, 5, White);
```

Modificar el gráfico anterior para que cada 100mSeg. avance hacia la derecha 10 píxeles.

```
for(;;){
    sprintf(str,"Variable i: %d", i);
    LCD_PrintText(10,32,str,Blue,White);
    i++;
    CoTimeDelay(0,0,0,100);

    LCD_Clear(Red);
    LCD_FillRectangle( 100+a, 100, 100, 20, Yellow);
    LCD_FillRectangle( 150+a, 80, 30, 20, Yellow);
    LCD_DrawRectangle( 155+a, 85, 20, 10, 2, Red);
    LCD_FillCircle( 120+a, 120, 10, Black);
    LCD_FillCircle( 180+a, 120, 10, Black);
    LCD_FillCircle( 120+a, 120, 5, White);
    LCD_FillCircle( 180+a, 120, 5, White);
    a = a+10;
}
```

2. Ejercicio 2

- Modificar **LCDGradientTask** para pintar degradados.
- Rellenar las pantalla con **barras verticales de 10 píxeles de ancho** (rectángulos), de manera que cada una de ellas contenga **cada uno de los 32 tonos de rojo, dejando las otras componentes a 0**.
- Esperar 500mSeg, volver a rellenar la pantalla con barras verticales verdes (64 tonos).
- Repetir para el color azul (32 tonos).
- Usar función **RGB565CONVERT (R, G , B)**.

```
void LCDGradientTask(void * parg){

    char str[40];
    char strP[40];
    LCD_Clear(Black);
    LCD_PrintText(10,224,"LCDGradientTask",White,Blue);
    int j;
    uint64_t i=0, v=0;

    //CoGetOSTime();

    for(;;){


        i=CoGetOSTime();
        for(j=0;j<32;j++){

            LCD_FillRectangle( 0 + 10*j, 0, 10, 240, RGB565CONVERT(j, 0, 0));

        }
        i=CoGetOSTime()-i;
        sprintf(str,"Tiempo pintar gradiente(ms): %d", i);
        v=(i*1000000)/76800;
        sprintf(strP,"Tiempo pintar pixel(ns): %d", v); //(i*1000)/76800
        LCD_PrintText(10,32,str,Blue,White);
        LCD_PrintText(10,64,strP,Blue,White);

        CoTimeDelay(0,0,0,500);



        i=CoGetOSTime();
        for(j=0;j<64;j++){

            LCD_FillRectangle( 0 + 10*j, 0, 10, 240, RGB565CONVERT(0, j, 0));

        }
        i=CoGetOSTime()-i;
        sprintf(str,"Tiempo pintar gradiente(ms): %d", i);
        v=(i*1000000)/76800;
        sprintf(strP,"Tiempo pintar pixel(ns): %d", v); //(i*1000)/76800
        LCD_PrintText(10,32,str,Blue,White);
        LCD_PrintText(10,64,strP,Blue,White);

        CoTimeDelay(0,0,0,500);



        i=CoGetOSTime();
        for(j=0;j<32;j++){

            LCD_FillRectangle( 0 + 10*j, 0, 10, 240, RGB565CONVERT(0, 0, j));

        }
        i=CoGetOSTime()-i;
        sprintf(str,"Tiempo pintar gradiente(ms): %d", i);
        v=(i*1000000)/76800;
        sprintf(strP,"Tiempo pintar pixel(ns): %d", v); //(i*1000)/76800
        LCD_PrintText(10,32,str,Blue,White);
        LCD_PrintText(10,64,strP,Blue,White);

        CoTimeDelay(0,0,0,500);
```

4. Driver del joystick analógico

`void Init_AnalogJoy (void)`

- Inicializa el GPIO, el ADC, y el DMA.

`uint16_t getAnalogJoy (uint8_t axxis)`

- Devuelve el valor del eje pasado como parámetro:
  - 0: Eje X
  - 1: Eje Y
- El valor devuelto está entre 0 y 4096 (12bits)

5. Ejercicio 3

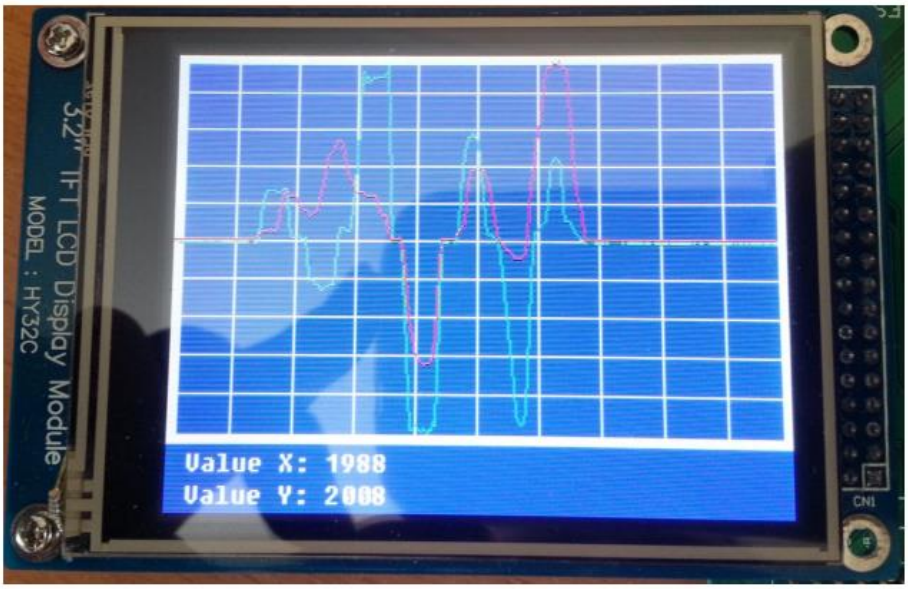
- **Pintar dos barras verticales con dos colores:**
  - Con una altura tota del 205 pix. y anchura de 10 pix.
  - Los colores serán rojo y verde, y la cantidad de cada uno debe ser proporcional a cada eje del joystick.
  - El valor de cada eje tiene un rango entre [0, 4095].
  - Si los dividimos entre 20 nos queda un rango entre [0, 205]

- **Área de pintado libre:**
  - Rellenar un rectángulo desde (0, 0) de 205 x 205 de color negro **antes de entrar en el bucle de la tarea**.
  - Pintar un círculo relleno de algún color en coordenadas proporcionales a los ejes del joystick, con un radio de 5 píxeles.
- Modificar la tarea para que el círculo del área de pintura siga la trayectoria real del joystick.

```
LCD_Clear(Blue);
//LCD_PrintText(10,224,"LCDDrawAreaTask",White,Blue);

char str[32];
char strP[32];
uint16_t x=0, y=0, xRect=0, yRect=0;
uint8_t xp = 0, yp = 1;


LCD_FillRectangle( 0, 0, 205, 205, RGB565CONVERT(0, 0, 0));

for(;;){
    CoTimeDelay(0,0,0,50);

    x = getAnalogJoy(xp);
    y = getAnalogJoy(yp);

    xRect = x/20;
    yRect = y/20;


    sprintf(str,"EJE X: %d  ", x);
    sprintf(strP,"EJE Y: %d  ", y);
    LCD_PrintText(10,207,str,Blue,White);
    LCD_PrintText(10,224,strP,Blue,White);


    LCD_FillRectangle( 260, 0, 10, 205, RGB565CONVERT(255, 0, 0));
    LCD_FillRectangle( 260, 205-xRect, 10, xRect, RGB565CONVERT(0, 255, 0));

    LCD_FillRectangle( 300, 0, 10, 205, RGB565CONVERT(255, 0, 0));
    LCD_FillRectangle( 300, 205-yRect, 10, yRect, RGB565CONVERT(0, 255, 0));
    //LCD_FillRectangle( x, y, ancho, alto, color)
    //LCD_FillCircle ( x, y, radio, color)

    LCD_FillCircle ( 205-xRect, 205-yRect, 5, Red);

}
```

6. Ejercicio 4

- Modificar la tarea **LCDScopeTask** para que simule un osciloscopio de dos canales.



- Visualizar el eje X del joystick analógico en el tiempo.
- Las coordenadas x de cada línea representarán el tiempo, y las coordenadas y han de ser proporcionales al valor del eje.
- Incrementar la coordenada x con el tiempo.
- Pintar una línea entre el valor actual del eje y el último valor leído en el instante de tiempo anterior.
- Usar un área de visualización de 320 x 205.

1. Modificar la tarea anterior para controlar el desborde de la pantalla.

2. Modificar la tarea anterior para muestre ambos ejes del joystick analógico.

3. Añadir un marco y una cuadrícula con 10 líneas horizontales y 10 líneas verticales homogéneamente repartidas.

```
LCD_Clear(Blue);

char str[32];
char strP[32];
uint16_t x=0, y=0, yAntigua = 0, xAntigua = 0;
uint16_t time=1;
uint8_t xp = 0, yp = 1;



LCD_FillRectangle( 0, 0, 320, 206, RGB565CONVERT(0, 0, 0));
xAntigua=getAnalogJoy(xp);
yAntigua=getAnalogJoy(yp);

int i;


LCD_FillRectangle( 0, 0, 10, 206, White);
LCD_FillRectangle( 310, 0, 10, 206, White);
for(i=10;i<310;i=i+30){
    LCD_DrawLine( i, 0, i, 205, White);
}

LCD_FillRectangle( 0, 0, 320, 13, White);
LCD_FillRectangle( 0, 192, 320, 13, White);
for(i=13;i<193;i=i+18){
        LCD_DrawLine( 0, i, 320, i, White);
}
```

```
for(;;){


    x = getAnalogJoy(xp);
    y = getAnalogJoy(yp);



    sprintf(str,"EJE X: %d  ", x);
    sprintf(strP,"EJE Y: %d  ", y);
    LCD_PrintText(10,207,str,Blue,White);
    LCD_PrintText(10,224,strP,Blue,White);



    LCD_DrawLine( time-1, 205-(yAntigua/20), time, 205-(y/20), Green);
    LCD_DrawLine( time-1, 205-(xAntigua/20), time, 205-(x/20), Red);

    yAntigua = y;
    xAntigua = x;
    time++;


    if(time == 320){
        LCD_FillRectangle( 0, 0, 320, 206, RGB565CONVERT(0, 0, 0));
        yAntigua=getAnalogJoy(yp);
        xAntigua=getAnalogJoy(xp);
        time=0;
        x=0;
        y=0;


        LCD_FillRectangle( 0, 0, 10, 206, White);
            LCD_FillRectangle( 310, 0, 10, 206, White);
            for(i=10;i<310;i=i+30){
                LCD_DrawLine( i, 0, i, 205, White);
            }

            LCD_FillRectangle( 0, 0, 320, 13, White);
            LCD_FillRectangle( 0, 192, 320, 13, White);
            for(i=13;i<193;i=i+18){
                    LCD_DrawLine( 0, i, 320, i, White);
            }

    }
    CoTimeDelay(0,0,0,10);


}
```