Librerías para TouchScreen
ILI9341 & XPT2046
Proyecto Hardware

Jorge Fernández Marín

@jofmar00

12 de junio de 2025

1. Introducción

En el siguiente documento vamos a abordar el diseño y uso de las librerías gráficas que usaremos para interactuar con la pantalla táctil `ILI9341` & `XPT2046`.
La pantalla que vamos a usar es un display bastante común en internet, dispone de 2 módulos distintos que se comunicarán por el protocolo `SPI` con nuestro microcontrolador, estos 2 módulos son los siguientes.

- `ILI9341`: Es el módulo display, se encarga de imprimir por pantalla.
- `XPT2046`: Es la parte del touch, que sirve para detectar una pulsación en la pantalla.

A partir de estos módulos se han creado unas librerías gráficas con el objetivo de poder trabajar con la pantalla táctil sin tener que preocuparnos del Hardware subyacente.

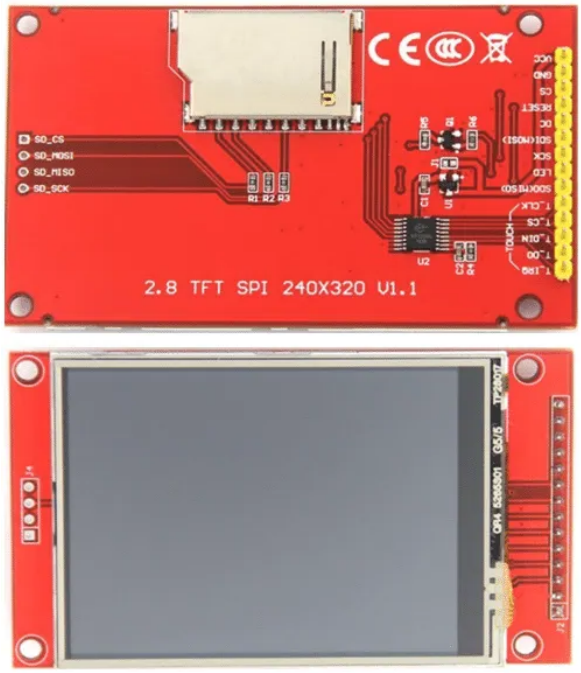

Figura 1: Pantalla ILI9341

2. Arquitectura Software

Las librerías gráficas del display pueden organizarse según 3 niveles de abstracción.

- `Hardware Abstraction Level`:
  Estos son los módulos que se comunican directamente con el Hardware subyacente mediante el protocolo de comunicación SPI. Para ello, utilizan las funciones dadas por el módulo `SPI.c`, que hemos recogido del Nordik SDK. Estos módulos exportan una serie de funcionalidades simples, como puede ser pintar un pixel en pantalla, con el objetivo de ser usadas por módulos de más alto nivel sin preocuparse del hardware.
  - `ILI9341.c`: Driver del display.
  - `XPT2046.c`: Driver del Touch.
- `Módulos a alto nivel`:
  Estos son los módulos con los que tendrá que trabajar el alumno, implementan funcionalidades complejas a partir de los módulos `HAL`.
  - `LCD_GFX.c`: Provee funciones gráficas para pintar diferentes tipos de objetos en la pantalla (cuadrados, círculos, texto, imágenes, etc). Para pintar un objeto en pantalla, usa un sistema de coordenadas `(x,y)`, que es variable en función de la rotación actual de la pantalla.
  - `LCD_TouchScreen.c`, provee funciones para detectar puntos de pulsación y presión en la pantalla. El módulo original `XPT2046`, devuelve cada una de las coordenadas `(x,y)`, en un rango `[0,4095]`, sin embargo, la pantalla tiene unas dimensiones de `320*240`px, por lo que para hacer la conversión, se definen una serie de valores, `MIN_X, MIN_Y, MAX_X y MAX_Y` definidos en el archivo `LCD_TouchScreen.h` que podeís alterar en caso de querer mejorar la calibración del módulo táctil en vuestra pantalla.
- `Demos Gráficas`:
  Son pequeñas demostraciones gráficas que sirven tanto comprobación de funcionamiento de la pantalla táctil como de ejemplo de uso de los módulos a alto nivel.
  - `LCD_GFX_test.c`: Demos gráficas
  - `LCD_TouchScreen_test.c`: Demos táctiles.

  En la sección 4. Ejemplo de uso se explica detenidamente el funcionamiento de la demo más complicada del módulo `LCD_TouchScreen_test.c`, que usa funciones de los dos módulos a alto nivel.

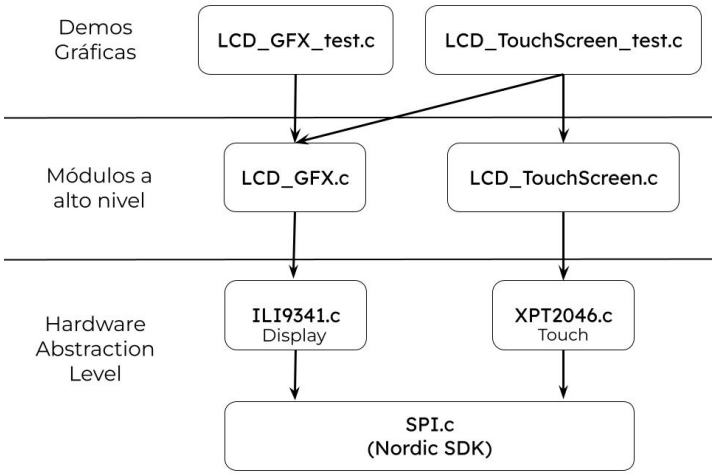

Figura 2: Arquitectura software de las librerias

3. Funciones que proporcionan las librerías

A continuación se muestran las funciones implementadas por los módulos `LCD_GFX` y `LCD_TouchScreen` junto sus cabeceras resumidas. Las cabeceras completas de las funciones se encuentran en los archivos `.h` de cada uno de los módulos.

3.1. LCD_GFX

```
// Inicializa la pantalla LCD.
void LCD_GFX_init(void);

// Establece la rotacion de la pantalla.
void LCD_GFX_setRotation(uint8_t dir);

// Dibuja una imagen en la pantalla a partir de un bitmap.
void LCD_GFX_drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t w, int16_t h,
↪  uint16_t color);

// Dibuja el contorno de un círculo en la pantalla LCD.
void LCD_GFX_drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color);

// Dibuja un círculo sólido (relleno) en la pantalla LCD.
void LCD_GFX_fillCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color);

// Dibuja una línea entre dos puntos en la pantalla LCD.
void LCD_GFX_drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color);

// Dibuja un rectángulo sin relleno en la pantalla LCD.
void LCD_GFX_drawRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);

// Dibuja un rectángulo sólido (relleno) en la pantalla LCD.
void LCD_GFX_fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);

// Dibuja una cadena de texto en la pantalla LCD.
void LCD_GFX_drawString(int16_t x, int16_t y, char* c, uint16_t color, uint16_t bg, uint8_t
↪  size);

// Dibuja un píxel en la pantalla LCD.
void LCD_GFX_drawPixel(int16_t x, int16_t y, uint16_t color);

// Llena toda la pantalla LCD con un color sólido.
void LCD_GFX_fillScreen(uint16_t color);
```

3.2. LCD_TouchScreen

```
// Inicializa el táctil.
void LCD_TouchScreen_init(void);

// Lee y procesa la posición de donde está siendo tocada la pantalla.
void LCD_TouchScreen_readPosition(uint16_t *x, uint16_t *y);

// Mide la presión del toque actual.
uint16_t LCD_TouchScreen_readPressure(void);

// Detecta presencia de toque en el panel.
bool LCD_TouchScreen_isTouched(void);
```

4. Ejemplo de uso

En este apartado vamos a analizar la demo gráfica `LCD_TouchScreen_test_paint`, que usa tanto funciones del módulo `LCD_GFX`, como `LCD_TouchScreen`. La demo consiste en una pantalla en blanco en la que puedes pintar con el lápiz táctil imitando el clasico paint de windows.
Pasemos con el código.

- Antes que nada limpiamos toda la pantalla en blanco, queremos nuestro lienzo en blanco.

```
LCD_GFX_fillScreen(WHITE);
```

- Vamos a añadir 2 cajas con texto, una servirá como título de la pantalla y la otra la usaremos como "botón"para salirnos de la demo y devolverle el flujo de programa al main.
  Esta demo trabaja con la pantalla en apaisado, por lo que para pintar el título de la pantalla con esa rotación deberemos indicarlo mediante `LCD_GFX_setRotation(1)`, el botón sin embargo está pintado con la rotación de la pantalla en vertical, también deberemos indicarlo.
  Se debe remarcar que no se tiene que cambiar la rotación cada vez que se escriba texto, sino cada vez que se cambie la rotación del texto. Es una buena práctica fijar una rotación al principio de cada función que vaya a escribir en pantalla puesto que no sabemos como se encuentra en ese momento variable que maneja el control de la rotación.

```
LCD_GFX_setRotation(0);
// "Salir" de texto negro sobre rojo en x=70, y=5 (rotación 0) con tamaño de
↪  letra 2
LCD_GFX_drawString(70, 5, "Salir", BLACK, RED, 2);

LCD_GFX_setRotation(1);
// "PINTA!" de texto magenta sobre blanco en x=120, y=10 (rotación 1) con
↪  tamaño de letra 3
LCD_GFX_drawString(120,10, "PINTA!", MAGENTA, WHITE, 3);
```

- Definimos variables para recoger los datos de lectura del panel táctil y una variable de control para ver si hamos tocado el botón de salir e iniciamos el bucle principal.

```
uint16_t x = 0, y = 0;
uint8_t terminado = 0;
while (!terminado) { ... }
```

- En cada iteración del bucle miramos si hay pulsación en la pantalla, y si es así, leemos las coordenadas del toque.

```
if (LCD_TouchScreen_isTouched()) {
    LCD_TouchScreen_readPosition(&x, &y);
    ...
}
```

- Si hemos presionado nuesto "botón", salimos de la demo, y si no, pintamos un cuadrado en la posición del toque.

```
// Terminar sesión pintado
if(x < 30 && (y > 70 && y <150)) {
    terminado = 1;
}
// Pintar normal
else {
    // Pintamos un rectangulo azul de 5*5px en la posicion del toque
    LCD_GFX_fillRect(x, y, 5, 5, BLUE);
}
```

El código al completo queda tal que así.

```
void LCD_TouchScreen_test_paint() {
    LCD_GFX_fillScreen(WHITE);
    LCD_GFX_setRotation(0);
    LCD_GFX_drawString(70, 5, "Salir", BLACK, RED, 2);
    LCD_GFX_setRotation(1);
    LCD_GFX_drawString(120,10, "PINTA!", MAGENTA, WHITE, 3);
    uint16_t x = 0, y = 0;
    uint8_t terminado = 0;
    while (!terminado) {
        if (LCD_TouchScreen_isTouched()) {
            LCD_TouchScreen_readPosition(&x, &y);
            // Terminar sesion pintado
            if(x < 30 && (y > 70 && y <150)) {
                terminado = 1;
            }
            // Pintar normal
            else {
                LCD_GFX_fillRect(x, y, 5, 5, BLUE);
            }
        }
    }
}
```

5. Repositorio y Soporte

El código fuente oficial de este proyecto está disponible en el siguiente repositorio de GitHub: https://github.com/jofmar00/ILI9341_XPT2046_Libraries.
Si tenéis alguna consulta, o habéis descubierto algún bug / error, podeis contactar con total confianza conmigo en `jofmar00@gmail.com` o cualquiera de mis redes sociales por el nombre de `@jofmar00` para mejorar el proyecto.
Mucha suerte!