DSBM - Laboratori 6: Incorporació de pantalla TFT a la Raspberry

Victor Expósito Griñán, Francesc Ruiz Tuyà

2 de juny de 2021

1 Introducció

L'objectiu d'aquesta última sessió de laboratori és comprendre com es pot incorporar una pantalla a la placa Raspberry Pi 2 per tal de mostrar informació per aquest dispositiu. La pantalla afegida és una pantalla TFT Proto com la de la Figura 1.

2 Esquema de les connexions

Primer de tot, hem realitzat un esquemàtic de com s'haurien de fer les connexions per tal que la pantalla funcioni i rebi dades de la Raspberry. Aquests dispositius s'han de comunicar per SPI, tal i com es veu a la Figura 2. Com que la pantalla té diversos ports de GND, és necessari connectar-los tots per que funcioni. També es necessita una connexió de LEDA i LEDK (que és bàsicament backlight) a 3,3 V i GND respectivament (LEDA necessita una resistència, detall especificat pel fabricant, així que hem decidit posar-li una de 220 Ohms).

Figura 1: Pantalla TFT Proto

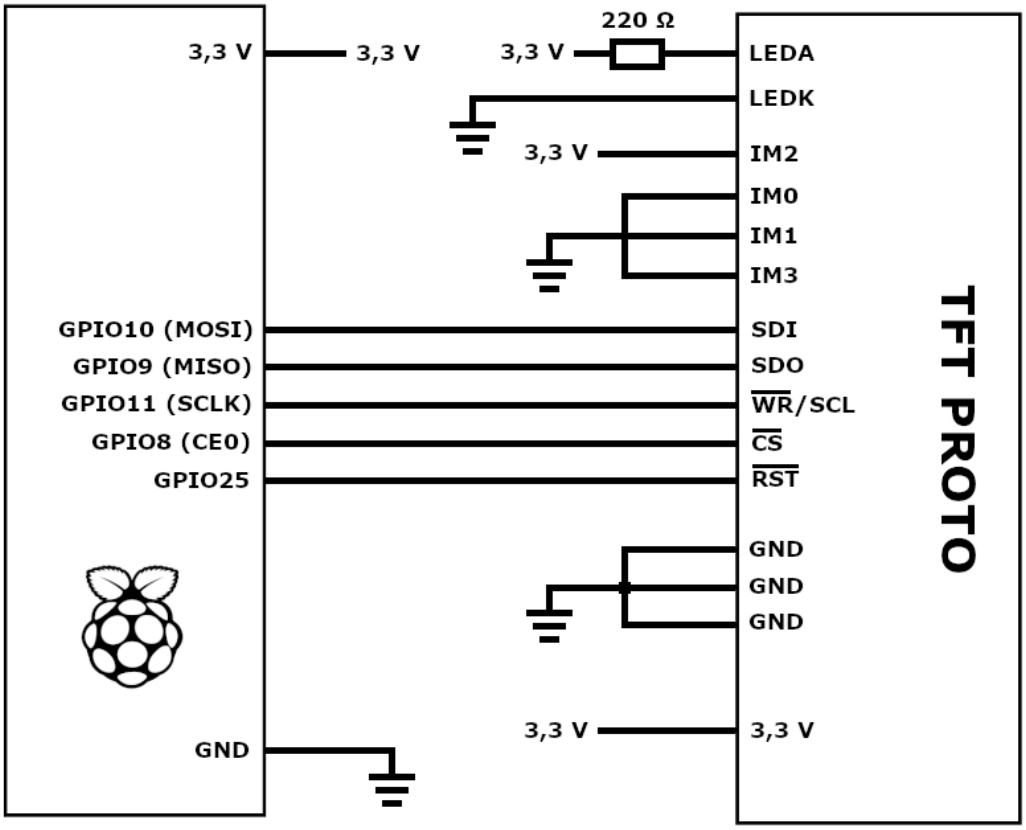

Figura 2: Esquemàtic de les connexions entre la Raspberry Pi 2 i la TFT Proto

3 Proves amb la pantalla

Un cop fetes les connexions pertinents, toca fer diferents tests per comprovar la funcionalitat de la pantalla. Per aconseguir aquest propòsit, hem escrit un joc de proves que es pot trobar a l'arxiu *driver.c*.

Per fer totes aquestes proves, abans hem de configurar els pins de la pantalla amb *Config_Pins()* i resetejar-la amb *SPI_TFT_Reset()*. En cas que es vulguin consultar els resultats de les diferents proves, anar a l'apartat 4.

3.1 Test 1

El primer test ha consistit en pintar un simple rectangle a la pantalla, el qual es mostra en un determinat color. La codificació de la pantalla és RGB565, així que els colors es codifiquen amb 16 bits.

```
void setBackground(int color)
{
    for (int i = 0; i < Size_X; ++i) {
        for (int j = 0; j < Size_Y; ++j) SPI_TFT_pixel(i, j, color);
    }
}

void printRectangle(int x1, int x2, int y1, int y2, int color)
{
    for (int i = x1; i < x2; ++i) {
        for (int j = y1; j < y2; ++j) SPI_TFT_pixel(i, j, color);
    }
}


// Test 1 - Print a set of pixels with a certain color
setBackground(0x0000);
printRectangle(100, 200, 80, 160, 0xFFFF);
```

La idea darrere la funció *setBackground* és establir tota la pantalla (320x240 píxels) a un color (0x0000 = negre en RGB565). Una vegada tota la pantalla està de color negre, el codi pinta un rectangle (*printRectangle*) amb els 4 vèrtexs a les coordenades (100, 80), (200, 80), (100, 160) i (200, 160), informació proporicionada pels paràmetres. L'últim paràmetre indica el color del rectangle (0xFFFF = blanc en RGB565). El resultat gràfic es pot veure la Figura 3.

3.2 Test 2

Un segon test sobre el printeig d'un conjunt de píxels passa per pintar una sèrie de línees verticals de colors seleccionables, usant els paràmetres de la funció *printVerticalLines*

```
void printVerticalLines(int color1, int color2)
{
    int i = 0, print1 = 1;
    int numLines = 8;
    for (int line = 0; line < numLines; ++line) {
        if (print1) {
            printRectangle(0, Size_X, i, i + 39, color1);
            print1 = 0;
        }
        else if (!print1) {
            printRectangle(0, Size_X, i, i + 39, color2);
            print1 = 1;
        }
        i += 40;
    }
}


// Test 2 - Print the screen with vertical alternative lines of 2 colors
printVerticalLines(0x4528, 0xFFFF);
```

Tal i com està escrit el codi, la pantalla queda dividida en 8 línees verticals de 40 píxels d'amplada cadascuna, pintades alternativament de verd (0x4528) i blanc (0xFFFF), com es mostra a la Figura 4.

3.3 Test 3

Un cop realitzats els testos sobre píxels per formar figures, hem decidit fer testos sobre caracters. Amb ajuda de l'arxiu *ascii5x7.h* i la funció *printCharacter* hem estat capaços d'interpretar i escriure un caracter a una posició concreta de la pantalla. En concret, el joc de prova el que fa és escriure el caracter '#' (codificat amb 5 bytes començant pel byte 15 dins l'array *Font5x7*) a la posició determinada. Com apunt, si printem el caracter amb *SPI_TFT_pixel(Size_X - X, Y, color)* és per una questió estètica, ja que volem que el text quedi escrit començant per dalt.

```
void printCharacter(int initPos, int posX, int posY, int color)
{
    char value, mask = 0b00000001;
    int X = posX, Y = posY;
    for (int pos = initPos; pos < initPos + 5; ++pos) {
        value = Font5x7[pos];
        for (int bit = 0; bit < 8; ++bit) {
            char printPixel = value & mask;
            if (printPixel) SPI_TFT_pixel(Size_X - X, Y, color);
            mask *= 2;
            ++X;
        }
        mask = 0b00000001;
        ++Y;
        X = posX;
    }
}
```

```
// Test 3 - Print a character with a certain color
setBackground(0x0000);
int initPos = 15;
printCharacter(initPos, 10, 10, 0xFFFF);
```

Novament, el resultat gràfic es pot veure a la Figura 5.

3.4 Test 4

Després d'haver printat un caracter amb èxit, el següent pas és printar una seqüència de caracters. És precisament aquest l'objectiu del test número 4. En aquesta part, es necessita un array que emmagatzemi les posicions inicials de cada caracter. El codi de prova printa la frase "*Hello World! :D*"a dalt de la pantalla, resultat que es pot consultar a la Figura 6.

```
// Test 4 - Print a phrase
setBackground(0x0000);
int numChars = 15;
int posX = 10, posY = 10;
int characters[15] = {40, 69, 76, 76, 79, 0, 55, 79, 82, 76, 68, 1, 0, 26, 36};
for (int i = 0; i < numChars; ++i) {
    printCharacter(characters[i] * 5, posX, posY, 0xFFFF);
    posY += 6;
}
```

3.5 Test 5

El següent pas dins del seguit de proves a realitzar és printar una imatge a la pantalla TFT. Per comoditat, hem usat imatges amb extensió bmp (bitmap), les quals codifiquen els colors dels píxels en format RGB888 (24 bits per píxel).

```
typedef unsigned int int32;
typedef short int16;
typedef unsigned char byte;

int readImage(const char *fileName,byte **pixels, int32 *width,
              int32 *height, int32 *bytesPerPixel)
{
        FILE *imageFile = fopen(fileName, "rb");
        int32 dataOffset;
        fseek(imageFile, DATA_OFFSET_OFFSET, SEEK_SET);
        int ret = fread(&dataOffset, 4, 1, imageFile);
        fseek(imageFile, WIDTH_OFFSET, SEEK_SET);
        ret = fread(width, 4, 1, imageFile);
        fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
        ret = fread(height, 4, 1, imageFile);
        int16 bitsPerPixel;
        fseek(imageFile, BITS_PER_PIXEL_OFFSET, SEEK_SET);
        ret = fread(&bitsPerPixel, 2, 1, imageFile);
        *bytesPerPixel = ((int32)bitsPerPixel) / 8;

        int paddedRowSize = (int)(4 * ceil((float)(*width) / 4.0f))*(*bytesPerPixel);
        int unpaddedRowSize = (*width)*(*bytesPerPixel);
        int totalSize = unpaddedRowSize*(*height);
        *pixels = (byte*)malloc(totalSize);
        int i = 0;
        byte *currentRowPointer = *pixels+((*height-1)*unpaddedRowSize);
        for (i = 0; i < *height; i++) {
                fseek(imageFile, dataOffset+(i*paddedRowSize), SEEK_SET);
                ret = fread(currentRowPointer, 1, unpaddedRowSize, imageFile);
                currentRowPointer -= unpaddedRowSize;
        }
        fclose(imageFile);
        return ret;
}

// Test 5 - Display an image 1
byte *pixels;
int32 width, height, bytesPerPixel;
readImage("tiger.bmp", &pixels, &width, &height, &bytesPerPixel);
int index = 0;
for (int i = 0; i < Size_X; ++i) {
    for (int j = 0; j < Size_Y; ++j) {
       int R = pixels[index + 2];
       int G = pixels[index + 1];
       int B = pixels[index];
       int pixelColor = (((R & 0xF8) << 8) + ((G & 0xFC) << 3) + (B >> 3));  // RGB888 to RGB565
       SPI_TFT_pixel(-1 * (i - Size_X - 1), j, pixelColor);
       index += 3;
    }
}
```

Amb la funció *readImage* es llegeix la capçalera bmp de la imatge anomenada *fileName*, la qual conté informació com el número de píxels, l'altura i amplada de la imatge, o els bytes per codificar el color de cada píxel (3, al nostre cas). Acte seguit, es llegeix la imatge, deixant el resultat com un array de bytes que representa el RGB de cada píxel (BGR, de fet, ja que bmp intercanvia aquests valors). Un cop agafats els valors de RGB corresponents a cada píxel, s'ha de fer la conversió de RGB888 a RGB565, la qual és molt senzilla, i ja es pot pintar a la pantalla amb *SPI_TFT_pixel*. La imatge escollida pel Test 5 és la fotografia de la Figura 7.

3.6 Test 6

Per últim, hem volgut printar una altre imatge per veure el resultat. El procediment ha estat el mateix que el del apartat anterior, i la imatge emprada ha estat la de la Figura 8.

```
// Test 6 - Display an image 2
byte *pixels;
int32 width, height, bytesPerPixel;
readImage("house.bmp", &pixels, &width, &height, &bytesPerPixel);
int index = 0;
for (int i = 0; i < Size_X; ++i) {
    for (int j = 0; j < Size_Y; ++j) {
       int R = pixels[index + 2];
       int G = pixels[index + 1];
       int B = pixels[index];
       int pixelColor = (((R & 0xF8) << 8) + ((G & 0xFC) << 3) + (B >> 3));  // RGB888 to RGB565
       SPI_TFT_pixel(-1 * (i - Size_X - 1), j, pixelColor);
       index += 3;
    }
}
```

4 Resultats

En aquest apartat final deixem imatges que representen els resultats obtinguts per les diferents proves.

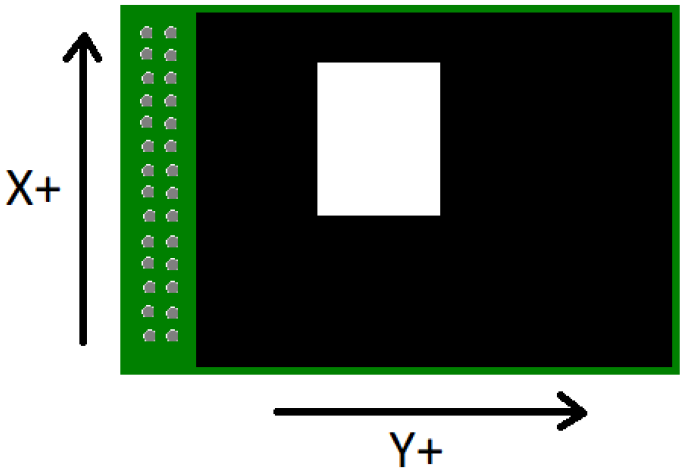

Figura 3: Resultat del Test 1 a la pantalla

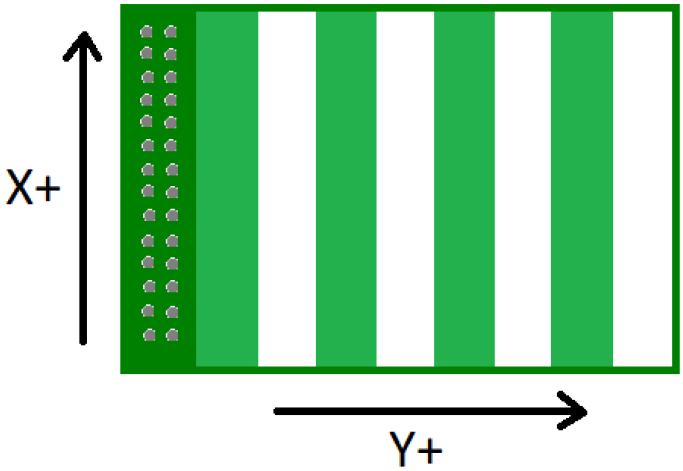

Figura 4: Resultat del Test 2 a la pantalla

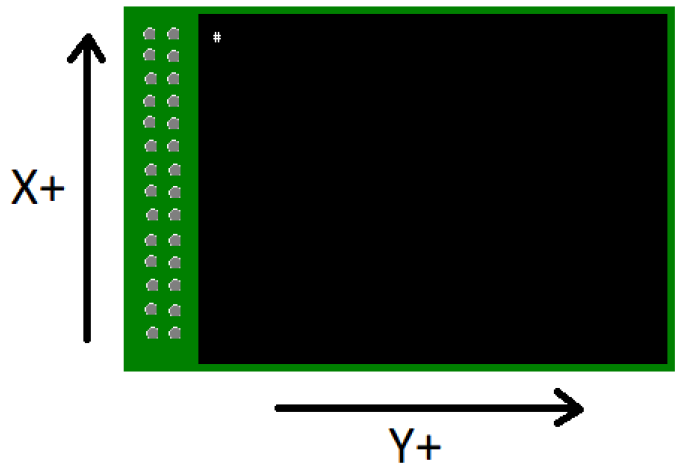

Figura 5: Resultat del Test 3 a la pantalla

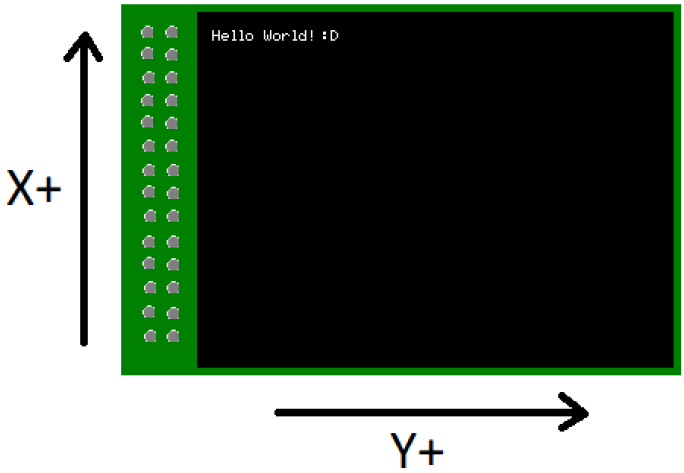

Figura 6: Resultat del Test 4 a la pantalla

Figura 7: Imatge del Test 5 a la pantalla

Figura 8: Imatge del Test 6 a la pantalla