



UNIVERSITAT POLITÈCNICA DE CATALUNYA

BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Estudi i desenvolupament d'un dispositiu portable per a la mesura de temperatura mitjançant infrarojos

Document:

[Annexos](#)

Autor/Autora:

[Arnau Gómez Sentís](#)

Director/Directora - Codirector/Codirectora:

[Miguel Delgado Prieto/ Didac Simon Garcia](#)

Titulació:

[Grau en Enginyeria Electrònica Industrial i
Automàtica](#)

Convocatòria:

[Primavera 2022](#)

TREBALL DE FÍD'ESTUDIS



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

**Estudi i desenvolupament d'un dispositiu
portable per a la mesura de temperatura
mitjançant infrarrojos**



Índex

ÍNDEX	3
1. CODIS PROGRAMACIÓ ARDUINO	4
1.1 CODI D'EXEMPLE DATASHEET DISPLAY TFT LCD	4
1.2 CODI IMPLEMENTAT EN FORMAT .TXT.....	16



1. Codis programació Arduino

1.1 Codi d'exemple datasheet display TFT LCD

```
*!  
* @file basicTest.ino  
* @brief Demonstrate various graphic painting effects  
* @n This demo supports Arduino Uno, Leonardo, Mega2560, FireBeetle-ESP32,  
FireBeetle-ESP8266, and FireBeetle-M0.  
* @copyright Copyright (c) 2010 DFRobot Co. Ltd (http://www.dfrobot.com)  
* @licence The MIT License (MIT)  
* @author [LuoYufeng] (yufeng.luo@dfrobot.com)  
* @version V0.1  
* @date 2020-01-07  
* @url https://github.com/DFRobot/DFRobot\_GDL  
*/  
  
#include "DFRobot_GDL.h"  
/*M0*/  
  
#if defined ARDUINO_SAM_ZERO  
#define TFT_DC 7  
#define TFT_CS 5  
#define TFT_RST 6  
/*ESP32 and ESP8266*/  
#elif defined(ESP32) || defined(ESP8266)  
#define TFT_DC D2  
#define TFT_CS D6  
#define TFT_RST D3  
/*AVR series mainboard*/  
#else  
#define TFT_DC 2  
#define TFT_CS 3  
#define TFT_RST 4  
#endif  
  
/**  
* @brief Constructor Constructor of hardware SPI communication  
* @param dc Command/data line pin for SPI communication
```



* @param cs Chip select pin for SPI communication

* @param rst reset pin of the screen

*/

```
DFRobot_ST7789_240x240_HW_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ST7789_240x320_HW_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ILI9341_240x320_HW_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ILI9488_320x480_HW_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
/* M0 mainboard DMA transfer */
//DFRobot_ST7789_240x240_DMA_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ST7789_240x320_DMA_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ILI9341_240x320_DMA_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
//DFRobot_ILI9488_320x480_DMA_SPI
screen(*dc=~/TFT_DC,*cs=~/TFT_CS,*rst=~/TFT_RST);
```

/*

*User-selectable macro definition color

*COLOR_RGB565_BLACK COLOR_RGB565_NAVY COLOR_RGB565_DGREEN
COLOR_RGB565_DCYAN

*COLOR_RGB565_MAROON COLOR_RGB565_PURPLE COLOR_RGB565_OLIVE
COLOR_RGB565_LGRAY

*COLOR_RGB565_DGRAY COLOR_RGB565_BLUE COLOR_RGB565_GREEN
COLOR_RGB565_CYAN

*COLOR_RGB565_RED COLOR_RGB565_MAGENTA COLOR_RGB565_YELLOW
COLOR_RGB565_ORANGE

*COLOR_RGB565_WHITE

*/

```
void setup() {
```

```
    Serial.begin(115200);
```

```
    screen.begin();
```

```
}
```

```

void loop(){
    testDrawPixel();
    testLine();
    testFastLines(COLOR_RGB565_PURPLE,COLOR_RGB565_YELLOW);
    testRects(COLOR_RGB565_BLACK,COLOR_RGB565_WHITE);
    testRoundRects();
    testCircles(24,COLOR_RGB565_BLUE);
    testTriangles(COLOR_RGB565_YELLOW);
    testPrint();

}

/* Test to draw a pixel*/
void testDrawPixel() {
    //Clear screen
    screen.fillScreen(COLOR_RGB565_BLACK);
    int x = 0;
    int y = screen.height();
    for(int i = 0; i <= screen.width()/2; i += 10){
        for (x = screen.width() - i; x >= i; x-=10 ){
            /*
             * @ brief draw a pixel
             * @ param x coordinate
             *      y coordinate
             * c pixel color
            */
            screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
            delay(10);
        }
        for (y = screen.height() - i; y >= i; y-=10){
            screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
            delay(10);
        }
        for (x = i; x <= screen.width() - i + 1; x+=10 ){

```



```
screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
delay(10);
}

for (y = i; y <= screen.height() - i + 1; y+=10){
    screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
    delay(10);
}
}

/*
 * Test to draw a line*/
void testLine(){
// 0x00FF is the color data in the format of RGB565
uint16_t color = 0x00FF;
screen.fillScreen(COLOR_RGB565_BLACK);
for (int16_t x=0; x < screen.width(); x+=6) {
/*
 * @ brief draw a line
 * @ param x0 The x-coordinate of the first vertex
 *      y0 The y-coordinate of the first vertex
 *      x1 The x-coordinate of the second vertex
 *      y1 The y-coordinate of the second vertex
 *      c line color
 */
screen.drawLine(/*x0=*/
screen.width()/*Screen width*///2,
/*y0=*/
screen.height()/*Screen height*///2, /*x1=*/x, /*y1=*/0, /*c=*/color+=0x0700);
}
for (int16_t y=0; y < screen.height(); y+=6) {
    screen.drawLine(screen.width()/2, screen.height()/2, screen.width(), y, color+=0x0700);
}

for (int16_t x = screen.width(); x >= 0; x-=6) {
    screen.drawLine(screen.width()/2, screen.height()/2, x,screen.height(), color+=0x0700);
}

for (int16_t y = screen.height(); y >= 0; y-=6) {
```



```
screen.drawLine(screen.width()/2, screen.height()/2, 0, y, color+=0x0700);
}

}

/* Test to fast draw line(need to set delay), only horizontal line and vertical line */
void testFastLines(uint16_t color1, uint16_t color2) {
    for (int16_t y=0; y < screen.height(); y+=4) {
        /*
         * @ brief draw a line
         * @ param x The x-coordinate of the first vertex
         *      y The y-coordinate of the first vertex
         *      w Length of line segment
         *      c line color
        */
        screen.drawFastHLine(/*x=*/0, /*y=*/y, /*w=*/screen.width(), /*c=*/color2);
        delay(10);
    }

    for(int16_t x=0; x < screen.width(); x+=3) {
        /*
         * @ brief draw a line
         * @ param x The x-coordinate of the first vertex
         *      y The y-coordinate of the first vertex
         *      h length of line segment
         *      c line color
        */
        screen.drawFastVLine(/*x=*/x, /*y=*/0, /*h=*/screen.height(), /*c=*/color1);
        delay(10);
    }
}

/* Test to draw a rectangle*/
void testRects(uint16_t color1, uint16_t color2) {
    screen.fillScreen(COLOR_RGB565_BLACK);
    int16_t x=screen.width()-12;
    for (; x > 100; x-=screen.width()/40) {
```

```

/*
 * @ brief draw a hollow rectangle
 * @ param x The x-coordinate of the vertex
 * @ param y The y-coordinate of the vertex
 * @ param w horizontal side length
 * @ param h longitudinal side length
 * @ param color Fill color, RGB color with 565 structure
 */
screen.drawRect(/*x=*/
screen.width()/2 -x/2, /*y=*/
screen.height()/2 -x/2 , /*w=*/
/*h=*/
x, /*color=*/
color2+=0x0F00);
delay(100);
}

/*
 * @ brief draw a filled rectangle
 * @ param x The x-coordinate of the vertex
 * @ param y The y-coordinate of the vertex
 * @ param w horizontal side length
 * @ param h longitudinal side length
 * @ param color Fill color, RGB color with 565 structure
 */
screen.fillRect(/*x=*/
screen.width()/2 -x/2, /*y=*/
screen.height()/2 -x/2 , /*w=*/
/*h=*/
x, /*color=*/
color2);
delay(100);
for(; x > 6; x-=screen.width()/40){
    screen.drawRect(screen.width()/2 -x/2, screen.height()/2 -x/2 , x, x, color1);
    delay(100);
}
}

/* Test to draw a rounded rectangle */
void testRoundRects() {
    screen.fillScreen(COLOR_RGB565_BLACK);
    // 0xF00F is the color data in the format of RGB565
    int color = 0xF00F;
    int i;
    int x = 0;
}

```



```
int y = 0;
int w = screen.width()-3;
int h = screen.height()-3;
for(i = 0 ; i <= 16; i+=2) {
/*
 * @ brief Draw a hollow rounded rectangle
 * @ param x0 The x-coordinate of the start vertex
 * @ param y0 The y-coordinate of the start vertex
 * @ param w horizontal side length
 * @ param h longitudinal side length
 * @ param radius Round corner radius
 * @ param color border color, 565 structure RGB color
*/
    screen.drawRoundRect(/*x0=*/x, /*y0=*/y, /*w=*/w, /*h=*/h, /*radius=*/20,
/*color=*/color);
    x+=5;
    y+=5;
    w-=10;
    h-=10;
    color+=0x0100;
    delay(50);
}
for(i = 0 ; i <= 16; i+=2) {
/*
 * @ brief Draw a filled and rounded rectangle
 * @ param x0 The x-coordinate of the start vertex
 * @ param y0 The y-coordinate of the start vertex
 * @ param w horizontal side length
 * @ param h longitudinal side length
 * @ param radius Round corner radius
 * @ param color Fill color, RGB color with 565 structure
*/
    screen.fillRoundRect(/*x0=*/x, /*y0=*/y, /*w=*/w, /*h=*/h, /*radius=*/10, /*color=*/color);
    x+=5;
    y+=5;
    w-=10;
    h-=10;
```



```
color+=0x0500;  
delay(50);  
}  
}  
  
/* Test to draw a circle */  
void testCircles(uint8_t radius, uint16_t color) {  
    screen.fillScreen(COLOR_RGB565_BLACK);  
    for (int16_t x=radius; x <=screen.width()-radius; x+=radius*2) {  
        for (int16_t y=radius; y <=screen.height()-radius; y+=radius*2) {  
            /*  
             * @ brief Draw a hollow circle  
             * @ param x0 The x-coordinate of the center point  
             * @ param y0 The y-coordinate of the center point  
             * @ param r radius  
             * @ param color Circle color, RGB color with 565 structure  
            */  
            screen.drawCircle(/*x0=*/x, /*y0=*/y, /*r=*/radius, /*color=*/color);  
            if(x == y ||x == -y ||x == y + 2*radius)  
            /*  
             * @ brief Draw a filled circle  
             * @ param x0 The x-coordinate of the center point  
             * @ param y0 The y-coordinate of the center point  
             * @ param r radius  
             * @ param color Fill color, RGB color with 565 structure  
            */  
            screen.fillCircle(/*x0=*/x, /*y0=*/y, /*r=*/radius, /*color=*/color);  
            color += 800;  
            delay(100);  
        }  
    }  
}  
  
/* Test to draw a triangle */  
void testTriangles(uint16_t color){  
    screen.fillScreen(COLOR_RGB565_BLACK);
```



```
for (int16_t i=0; i <=screen.width(); i+=24)
/*
 * @ brief Draw a hollow triangle
 * @ param x0 The x-coordinate of the start vertex
 * @ param y0 The y-coordinate of the start vertex
 * @ param x1 The x-coordinate of the second vertex
 * @ param y1 The y-coordinate of the second vertex
 * @ param x2 The x-coordinate of the third vertex
 * @ param y2 The y-coordinate of the third vertex
 * @ param color border color, 565 structure RGB color
 */
screen.drawTriangle(/*x0=*/i,/*y0=*/0,/*x1=*/0,/*y1=*/screen.height()-
i,/*x2=*/screen.width()-i,/*y2=*/screen.height(), /*color=*/color);

for (int16_t i=0; i <screen.width(); i+=24)
    screen.drawTriangle(screen.width(),i*4/3,0,screen.height()-i*4/3,i,0, color);

for (int16_t i=0; i <screen.width(); i+=24)
    screen.drawTriangle(screen.width(),i*4/3,i,0,screen.width()-i,screen.height(), color);

color = COLOR_RGB565_RED;
for (int16_t i=0; i <=screen.width(); i+=24)
/*
 * @ brief Draw a filled triangle
 * @ param x0 The x-coordinate of the start vertex
 * @ param y0 The y-coordinate of the start vertex
 * @ param x1 The x-coordinate of the second vertex
 * @ param y1 The y-coordinate of the second vertex
 * @ param x2 The x-coordinate of the third vertex
 * @ param y2 The y-coordinate of the third vertex
 * @ param color Fill color, RGB color with 565 structure
 */
screen.fillTriangle(/*x0=*/i,/*y0=*/0,/*x1=*/0,/*y1=*/screen.height()-
i,/*x2=*/screen.width()-i,/*y2=*/screen.height(), /*color=*/color+=100);

for (int16_t i=0; i <screen.width(); i+=24)
```



```
screen.fillTriangle(screen.width(),i*4/3,0,screen.height()-i*4/3,i,0, color+=100);

for (int16_t i=0; i <screen.width(); i+=24)
    screen.fillTriangle(screen.width(),i*4/3,i,0,screen.width()-i,screen.height(), color+=100);
}

void testPrint() {
    // 0x00FF is the color data in the format of RGB565
    int16_t color = 0x00FF;
    // Set text wrapping mode
    // true = Text word wrap, false = No word wrap
    screen.setTextWrap(false);
    //Fill color, RGB color with 565 structure
    screen.fillScreen(COLOR_RGB565_BLACK);

    //Set the coordinate position x = 0, y = 50
    screen.setCursor(0, 50);
    //Set the text color; this is a changeable value
    screen.setTextColor(color+=0x3000);
    //Set text size to 0
    screen.setTextSize(0);
    //Output text
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 1
    screen.setTextSize(1);
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 2
    screen.setTextSize(2);
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 3
```



```
screen.setTextSize(3);
screen.println("Hello World!");

screen.setTextColor(color+=0x3000);
//Set text size to 4
screen.setTextSize(4);
screen.println("Hello!");
//Set text size to 5
screen.setTextSize(5);
screen.print("Hello!");
delay(2000);

//Set coordinate position x = 0, y = 0
screen.setCursor(0, 0);
//Fill color, RGB color with 565 structure
screen.fillScreen(COLOR_RGB565_BLACK);
screen.setTextSize(2);
screen.setTextColor(color+=0x3000);
screen.print("a = ");

screen.setTextColor(color+=0x3000);
int a = 1234;
screen.println(a, 1);
screen.setTextColor(color+=0x3000);
screen.print(8675309, HEX);
screen.println("this is HEX!");
screen.println("");

screen.setTextColor(color+=0x0F00);
screen.println("running for: ");
screen.setTextColor(color+=0x0F00);
//Output time in millisecond
screen.print(millis());
screen.setTextColor(color+=0x0F00);
screen.println("/1000 seconds.");
```



```
char *text = "Hi DFRobot!";
screen.setTextColor(color+=0x0F00);
screen.setTextWrap(true);
screen.setTextSize(3);
screen.println(text);
//screen.setFonts((const gdl_Font_t *)SIMKAIFont18ptBitmaps);
screen.println(text);
delay(2000);
}
```



1.2 Codi implementat en format .txt

```
//MN Maker
//Laser Temp Gun
//10.6.19
#include "DFRobot_GDL.h"
/*M0*/
#if defined ARDUINO_SAM_ZERO
#define TFT_DC 7
#define TFT_CS 5
#define TFT_RST 6
/*ESP32 and ESP8266*/
#elif defined(ESP32) || defined(ESP8266)
#define TFT_DC D2
#define TFT_CS D6
#define TFT_RST D3
/*AVR series mainboard*/
#else
#define TFT_DC 2
#define TFT_CS 3
#define TFT_RST 4
#endif
/**
 * @brief Constructor Constructor of hardware SPI communication
 * @param dc Command/data line pin for SPI communication
 * @param cs Chip select pin for SPI communication
 * @param rst reset pin of the screen
 */
DFRobot_ST7789_240x240_HW_SPI
screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ST7789_240x320_HW_SPI
screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9341_240x320_HW_SPI
screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9488_320x480_HW_SPI
screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
/* M0 mainboard DMA transfer */
//DFRobot_ST7789_240x240_DMA_SPI
screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
```



```
//DFRobot_ST7789_240x320_DMA_SPI
screen(*dc=*/TFT_DC,*cs=*/TFT_CS,*rst=*/TFT_RST);
//DFRobot_ILI9341_240x320_DMA_SPI
screen(*dc=*/TFT_DC,*cs=*/TFT_CS,*rst=*/TFT_RST);
//DFRobot_ILI9488_320x480_DMA_SPI
screen(*dc=*/TFT_DC,*cs=*/TFT_CS,*rst=*/TFT_RST);

/*
 *User-selectable macro definition color
 *COLOR_RGB565_BLACK    COLOR_RGB565_NAVY      COLOR_RGB565_DGREEN
 COLOR_RGB565_DCYAN
 *COLOR_RGB565_MAROON    COLOR_RGB565_PURPLE   COLOR_RGB565_OLIVE
 COLOR_RGB565_LGRAY
 *COLOR_RGB565_DGRAY    COLOR_RGB565_BLUE    COLOR_RGB565_GREEN
 COLOR_RGB565_CYAN
 *COLOR_RGB565_RED      COLOR_RGB565_MAGENTA COLOR_RGB565_YELLOW
 COLOR_RGB565_ORANGE
 *COLOR_RGB565_WHITE

#include <Wire.h>
#include <Adafruit mlx90614.h>

const int Laser_Pin=5; //Laser Pin
int buttonState = 0;
const int buttonPin = 6; // the number of the pushbutton pin

Adafruit mlx90614 mlx = Adafruit mlx90614();

void setup() {
  Serial.begin(9600);
  Serial.println("Adafruit mlx90614 test");

  pinMode(Laser_Pin,OUTPUT);
  pinMode(buttonPin, INPUT);
```



```
screen.begin();
InitPrint();

mlx.begin();

}

void loop() {

buttonState = digitalRead(buttonPin);
Serial.println(buttonState);

Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
Serial.print("*C\ltObject = "); Serial.print(mlx.readObjectTempC()); Serial.println("*C");

// check if the pushbutton is pressed. If it is, the buttonState is HIGH:
if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(Laser_Pin, HIGH);

    PrintValue();

}

else {
    // turn LED off:
    digitalWrite(Laser_Pin, LOW);

    PrintValueOff();

}

Serial.println();
delay(500);

}
```



```
void InitPrint() {  
    // 0x00FF is the color data in the format of RGB565  
    int16_t color = 0x00FF;  
    // Set text wrapping mode  
    // true = Text word wrap, false = No word wrap  
    screen.setTextWrap(false);  
    //Fill color, RGB color with 565 structure  
    screen.fillScreen(COLOR_RGB565_BLACK);  
  
    //Set coordinate position x = 0, y = 0  
    //screen.setCursor(0, 0);  
  
    screen.setTextColor(color+=0x0F00);  
    screen.setTextWrap(true);  
    screen.setTextSize(5);  
}  
  
void PrintValue() {  
    int w = screen.width();  
    screen.fillRect(30,100,w,35, COLOR_RGB565_BLACK);  
    screen.setCursor(30, 100);  
    screen.print(mlx.readObjectTempC());  
    screen.print(" C");  
  
}  
void PrintValueOff() {  
    int w = screen.width();  
    screen.fillRect(30,100,w,35, COLOR_RGB565_BLACK);  
    screen.setCursor(30, 100);  
    screen.print("*****");  
  
}
```