CCADET

CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLÓGICO
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Manual de uso

Configuración de una pantalla sensible al tacto para Arduino

Miguel Angel Bañuelos Saucedo

Febrero 2016

Proyecto de desarrollo tecnológico

Financiamiento interno

Título: Configuración de una pantalla sensible al tacto para Arduino

Autor: Miguel Angel Bañuelos Saucedo

Afiliación: Centro de Ciencias Aplicadas y Desarrollo Tecnológico, Universidad Nacional Autónoma de México

Resumen

La interfaz humano-máquina es uno de los subsistemas importantes en un aparato electrónico, debido a que su correcto diseño proporciona características de facilidad de uso y robustez. Además, puede formar parte importante de la estética del producto. Tradicionalmente, los equipos desarrollados en el CCADET (Centro de Ciencias Aplicadas y Desarrollo Tecnológico) basan su interfaz de operación en alguno de dos sistemas: en una computadora personal, o en botones y un display LCD (del inglés, *Liquid Crystal Display*). En un ambiente donde cotidianamente utilizamos interfaces sensibles al tacto (en inglés, *touch-screen*), ya sea en teléfonos celulares o cajeros automáticos, resulta conveniente tratar de dotar de esa tecnología a los instrumentos electrónicos que se desarrollan. Una pantalla sensible al tacto, permite desplegar información en color (aunque también hay versiones monocromáticas), dando la libertad al diseñador de proponer una interfaz dinámica, apoyada por íconos, gráficas, o animaciones. En este reporte, se explica el proceso de configuración de una pantalla sensible al tacto, y se muestra un ejemplo de interfaz de ajuste de parámetros. El ejemplo, del cual se detalla la programación en lenguaje Sketch, puede utilizarse como base para desarrollar un control de temperatura o cualquier otra variable. Se asume por parte del lector un nivel básico de conocimiento de este lenguaje. Con fines demostrativos se utiliza una tarjeta Arduino UNO, pero los programas son compatibles con otro tipo de tarjetas de la familia Arduino.

Índice

Resumen 1
Nomenclatura 3
Introducción 4
1 Características del hardware 5
1.1 Arduino UNO 5
1.2 Pantalla de cristal líquido sensible al tacto 6
1.3 Instalación de las bibliotecas de manejo del display 7
1.4 Despliegue de una imagen 8
2 Creación de una aplicación utilizando las características táctiles de la pantalla 8
Conclusiones 10
Anexo A. Listado del programa 11
ANEXO B. Tabla de definición de colores 15
Referencias Bibliográficas 16

Nomenclatura

Símbolo	*Significado*
AREF	*Referencia externa para el convertidor analógico-digital*
BMP	*Bit Map*
CD	*Corriente Directa*
EEPROM	*Electrically Erasable Programmable Read Only Memory*
GND	*Tierra (Ground)*
LED	*Diodo emisor de luz*
ON	*Encendido*
PWM	*Pulse Width Modulation*
RGB	*Red, Green, and Blue*
RX	*Recepción*
SD	*Secure Digital*
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
TFT	*Thin-Film Transistor*
TX	*Transmisión*
UNO	*No representa ningún acrónimo*
USB	*Universal Serial Bus*

Introducción

Existen dos tecnologías de amplio uso para el funcionamiento de pantallas sensibles al tacto: resistiva y capacitiva. Su nombre indica el principio físico en que basan su operación. Al tocar la superficie de la pantalla, es posible determinar la posición del contacto mediante la medición de un cambio en la resistencia o la capacitancia de una o más películas que recubren a la pantalla. La tecnología resistiva es más económica, mientras que la tecnología capacitiva permite elaborar pantallas que presentan una imagen más clara (Downs, 2005). La descripción detallada del funcionamiento de este tipo de pantallas queda fuera del alcance de este reporte, sin embargo, el lector puede encontrar más información en (Walker, 2012).

La tecnología que se seleccionó para desarrollar el ejemplo de aplicación de una pantalla sensible al tacto fue una tarjeta Arduino. Las tarjetas Arduino cuentan con un ambiente de programación de uso libre, multiplataforma (Windows, Linux y MAC OS X), y con una arquitectura de conexiones periféricas estandarizada, la cual permite que los fabricantes puedan desarrollar módulos periféricos compatibles con la familia Arduino. En particular se utilizó la tarjeta Arduino UNO y el módulo 2.8" TFT *Touch Shield for Arduino with Resistive Touch Screen* de la empresa Adafruit. El módulo TFT se inserta directamente en los puertos de conexión de la tarjeta Arduino.

El fabricante proporciona las bibliotecas para controlar la pantalla mediante el Arduino. De esta manera, se simplifican tareas tales como desplegar texto, o dibujar puntos, líneas, círculos, cuadrados y triángulos; ya que para ello solo basta con utilizar una función con los parámetros adecuados. La pantalla permite la creación de una interfaz humano/máquina más poderosa a un costo moderado, pues la misma se puede conseguir en México por alrededor de $800 pesos.

1 Características del hardware

1.1 Arduino UNO

Como se mencionó en la introducción el sistema que se propuso para la demostración del uso de una pantalla sensible al tacto fue la plataforma Arduino UNO. Esta tarjeta se puede conseguir en el mercado nacional a precios que oscilan entre los $230 pesos y los $420 pesos, dependiendo del proveedor, y de si el componente se trata de un circuito original o un clon. Las tarjetas originales son hechas en Italia, y se distinguen por una mejor calidad en la serigrafía. No se ha encontrado, sin embargo, alguna diferencia operativa con respecto a las versiones clonadas.

El Arduino UNO es un módulo basado en un microcontrolador de 8 bits ATmega328 (Atmel, 2014), que opera a 5 V. Otros módulos de la familia Arduino pueden utilizar otro tipo de microcontroladores (incluyendo tipo ARM™) y operar a 3.3 V, por ejemplo. Es importante notar la diferencia en voltaje de operación, pues algunos módulos periféricos (denominados Shields), pueden no ser compatibles con ambos voltajes.

En la Figura 1 se muestra la vista superior de la tarjeta Arduino UNO. Ésta incluye un controlador de comunicaciones USB, y es a través de un puerto USB que se puede efectuar la programación del microcontrolador (ver Figura 1).Las características principales de esta tarjeta se enlistan en la Tabla 1. Se observa que los pines pueden suministrar/drenar corrientes de hasta 40 mA, sin embargo, no debe superarse un total máximo de 200 mA (Atmel, 2014).

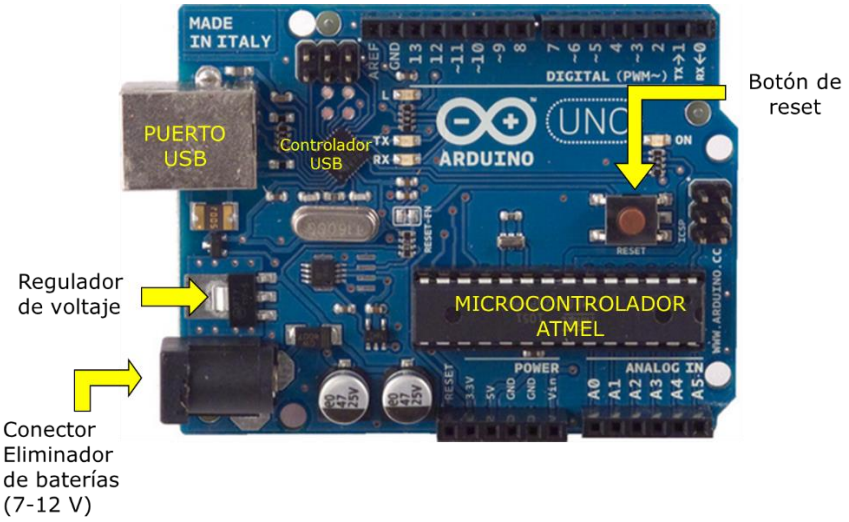

Figura 1. Vista superior de una tarjeta Arduino UNO, donde se muestra la ubicación de algunos de los componentes principales.

Para facilitar su uso, los pines designados para operaciones digitales se encuentran en un lado de la tarjeta y tienen la numeración del 0 al 13, mientras que los pines que tienen capacidades para funcionar como entradas analógicas están ubicados en el lado opuesto de la tarjeta. Adicionalmente la tarjeta presenta conexiones de 3.3 V, 5 V y tierra (GND) para alimentar circuitos externos. También existe una conexión para proporcionar una referencia de voltaje externa (AREF) para el convertidor analógico-digital del microcontrolador. Existen cuatro LED, uno para mostrar que la tarjeta se encuentra encendida (tiene la etiqueta ON), dos para señalizar la transmisión serial (con etiquetas RX, para la recepción, y TX, para la transmisión), y un último de propósito general conectado al pin 13 digital (con etiqueta L). La tarjeta puede tomar su voltaje de alimentación del puerto USB, o bien por medio de un conector a un eliminador de baterías u otra fuente de voltaje continuo entre 7 y 12 V.

Microcontrolador	ATmega328
Voltaje de operación	5V
Voltaje de suministro (recomendado)	7-12V
Voltaje de suministro (límites)	6-20V
Pines de entrada/salida	14 (de los cuales 6 proveen salida PWM)
Pines de entrada analógica	6 (10 bits, ± 2 LSB exactitud absoluta)
Corriente CD por pin E/S	40 mA
Corriente CD por pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) (0.5 KB para el cargador (bootloader)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Frecuencia del reloj	16 MHz
Longitud/Ancho	68.6 mm/53.4 mm
Peso	25 g

Tabla 1. Características principales de la tarjeta Arduino UNO.

1.2 Pantalla de cristal líquido sensible al tacto

La pantalla que se seleccionó fue el modelo de 2.8 pulgadas TFT con pantalla sensible al tacto de tecnología resistiva de la empresa Adafruit (Product ID: 1651) (que se adquirió en el mercado nacional a un costo aproximado de $800 pesos (agosto de 2015). Cuenta con una resolución de 320 x 240 pixeles con 18 bits de color (262,000 tonos). El control se realiza por medio de una interfaz SPI, requiriendo 5 pines SPI para el control de la pantalla y un pin SPI para el controlador de la operación táctil. Tiene además una ranura que acepta tarjetas microSD, de la cual es posible leer imágenes en formato BMP (ver Figura 2). La conexión de la pantalla ocupa todos los puertos de la tarjeta Arduino UNO; sin embargo, no utiliza la totalidad de ellos. En caso que se requiera utilizar los puertos libres del Arduino, será necesario introducir conectores adicionales.

Figura 2. Pantalla de cristal líquido montada sobre la tarjeta Arduino UNO.

1.3 Instalación de las bibliotecas de manejo del display

Para poder manejar el display, se requiere instalar las bibliotecas de manejo del mismo. Se necesitan las bibliotecas Adafruit-GFX-Library-master.zip, Adafruit_ILI9341-master.zip y Adafruit_STMPE610-master.zip, las cuales corresponden a las funciones gráficas, al controlador del display y al controlador de la pantalla resistiva (sensible al tacto), respectivamente (Burgess, 2016). Las bibliotecas se instalan usando el menú que se muestra en la Figura 3. Una vez instaladas, se puede probar el funcionamiento del display utilizando el ejemplo que podemos hallar mediante el menú mostrado en la Figura 4.

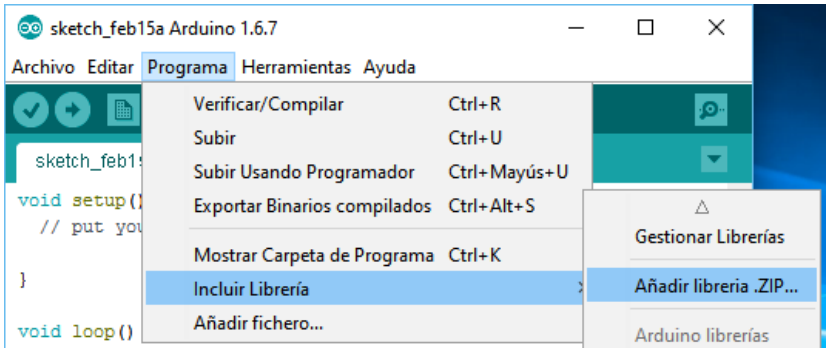

Figura 3. Ventana que muestra la opción de instalación de las bibliotecas (librerías).

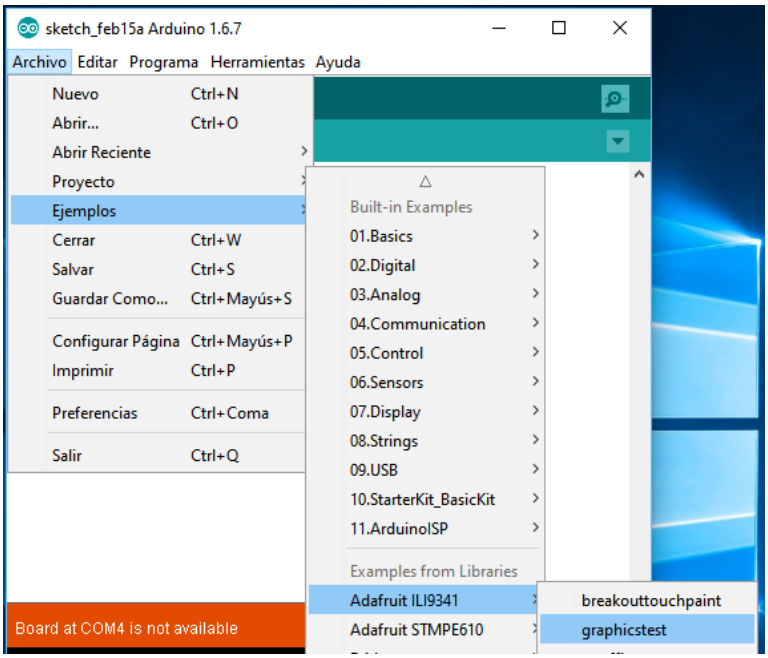

Figura 4. Menú para cargar el ejemplo de manejo del display LCD gráfico.

El programa de ejemplo, mostrará algunos textos y después una serie de gráficas, como la que se muestra en la Figura 5.

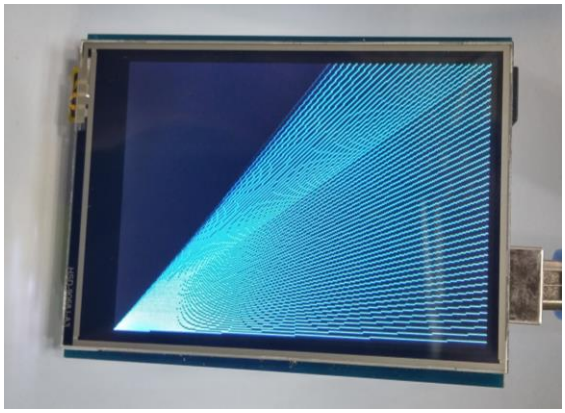

Figura 5. Funcionamiento del programa de ejemplo del display gráfico.

1.4 Despliegue de una imagen

Se puede encontrar un programa ejemplo de cómo desplegar una imagen bitmap en el menú de ejemplos y seleccionando **spitftbmp**. Para cargar una imagen guardada en la memoria microSD que se puede instalar en el display, basta con cambiar la línea 45 del ejemplo para ajustarla al nombre de nuestro archivo: *bmpDraw("Logopuma.bmp", 0, 0)*; donde es conveniente que el archivo BMP contenga una imagen de 320 x 240 pixeles. En la Figura 2, se muestra el resultado de la ejecución del programa modificado.

2 Creación de una aplicación utilizando las características táctiles de la pantalla

Ahora comentaremos el procedimiento para realizar una aplicación de ajuste de un parámetro. En el desarrollo de un sistema electrónico este parámetro puede ser la temperatura, la velocidad, o cualquier otra variable que se esté controlando o configurando. Para efectuar el ajuste de la variable se usarán dos botones uno para incrementar su valor y otro para disminuirlo. Para crear los botones en la pantalla, se utilizan las funciones de generación de polígonos que se incluyen en las bibliotecas, las cuales incluyen, triángulos y rectángulos, entre otras. La ubicación de los pixeles se determina por las coordenadas X y Y, según se ilustra en la Figura 6. Se puede encontrar información más detallada sobre las funciones gráficas en (Burgess, 2014).

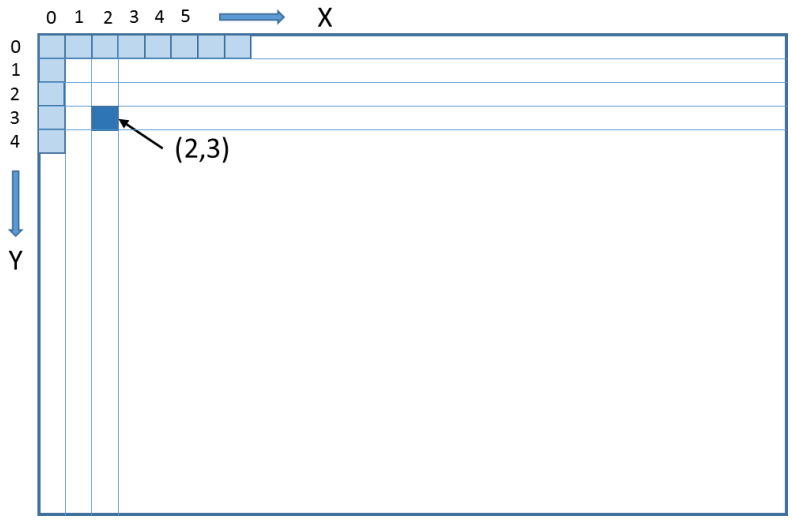

Figura 6. Definición de las coordenadas de ubicación de los pixeles en la pantalla.

De acuerdo a la matriz de pixeles de la pantalla utilizada (320 x 240), se define la ubicación de los botones según se muestra en la Figura 7. Los rectángulos se generan dando las coordenadas de un vértice, más el ancho y alto. Los triángulos se dibujan proporcionando las coordenadas de los tres vértices. El rectángulo negro que envuelve a cada flecha consiste en la definición del área que se va a definir para la detección de la presión en la pantalla.

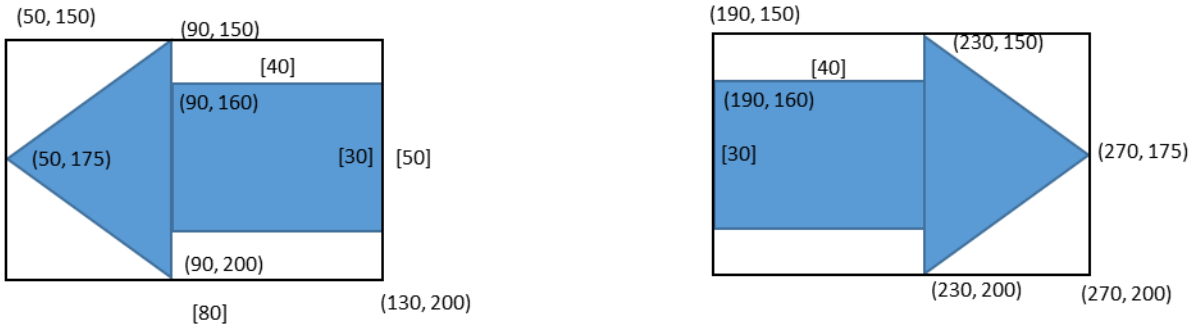

Figura 7. Parámetros de creación de las figuras asociadas a los botones.

Para darle una mejor presentación a la interfaz gráfica se le añaden sombras a los botones, cuyas coordenadas se anotan en la Figura 8. Las áreas de las figuras que constituyen las sombras son ligeramente más grandes que las flechas, y se encuentran desplazadas diagonalmente hacia abajo. Durante la operación de la interfaz gráfica, al presionar el botón, el botón original se borra y la sombra se colorea para reemplazar al botón; de esta manera se crea el efecto de que el botón ha sido presionado. Al soltarse el botón el procedimiento se invierte. Al programa se le añaden retrasos para evitar que la variable pueda cambiar más rápido que la velocidad de despliegue de los datos.

Figura 8. Parámetros para la generación de las sombras de las flechas.

Conclusiones

El desarrollo de una interfaz gráfica para una pantalla LCD sensible al tacto se simplifica en gran medida por la existencia de bibliotecas de funciones para la tarjeta Arduino. De esta manera, el trabajo se centra en diseñar los elementos gráficos de la interfaz. Es posible que para definir las dimensiones adecuadas y los efectos deseados de movimiento, se requieran efectuar algunos ensayos. En la Figura 9, se muestra un ejemplo del funcionamiento de la interfaz gráfica sensible al tacto

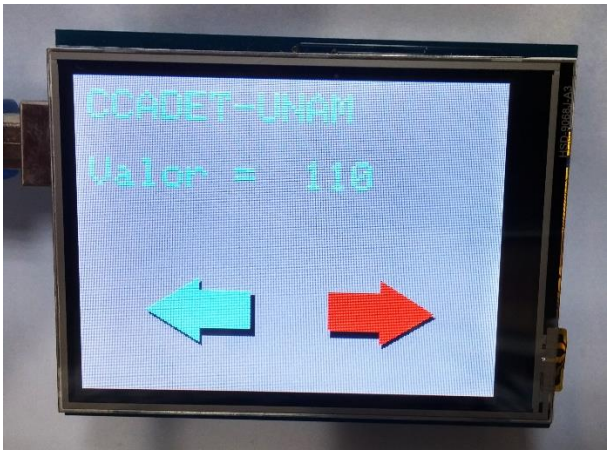

Figura 9. Fotografía de la interfaz gráfica desarrollada.

Anexo A. Listado del programa

El programa que se desarrolló está basado en el ejemplo incluido en las bibliotecas, la versión tal y como fue modificada se muestra a continuación.

```
/***************************************************
  This is our GFX example for the Adafruit ILI9341 Breakout and Shield
  ----> http://www.adafruit.com/products/1651

  Check out the links above for our tutorials and wiring diagrams
  These displays use SPI to communicate, 4 or 5 pins are required to
  interface (RST is optional)
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>

// This is calibration data for the raw touch data to the screen coordinates
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000

#define CYANBUTTON_X1 50       //Define ventana de detección de toque
#define CYANBUTTON_Y1 150
#define CYANBUTTON_X2 130
#define CYANBUTTON_Y2 200

#define REDBUTTON_X1 190      //Define ventana de detección de toque
#define REDBUTTON_Y1 150
#define REDBUTTON_X2 270
#define REDBUTTON_Y2 200

#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
#define TFT_CS 10
#define TFT_DC 9
```

```
// Use hardware SPI (on Uno, #13, #12, #11) and the above for CS/DC
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
// If using the breakout, change pins as desired
//Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK, TFT_RST,
TFT_MISO);

boolean leeTouch = false;
int valor = 100;

void setup() {
  Serial.begin(9600);      // Inicializa comunicación serial
  Serial.println("Hola CCADET");

// La comunicación serial se utiliza para poder verificar que arrancó el programa
  tft.begin();     // Inicializa el display
   if (!ts.begin()) {
    Serial.println("Unable to start touchscreen.");    // envía aviso en caso de error
  }
  else {
    Serial.println("Touchscreen started.");    // envía aviso en caso de éxito
  }

  tft.fillScreen(ILI9341_DARKGREY);
  tft.setRotation(1);  // Pantalla horizontal
  tft.setCursor(10, 10);  // Posición inicial del cursor
  tft.setTextColor(ILI9341_CYAN);    // Selección de color de acuerdo a las bibliotecas
  tft.setTextSize(3);      // Ajuste de tamaño
  tft.println("CCADET-UNAM");    // Mensaje de texto. Título de la interfaz

  tft.setCursor(10, 60);
  tft.print("Valor = ");
  tft.setCursor(170, 60);
  tft.println(valor);      // Despliega el valor

  // Se dibujan dos flechas (izq. y der.) para controlar una variable

  tft.fillRect(92, 163, 42, 33, ILI9341_BLACK);  //  Dibuja sombra rec izq
  tft.fillTriangle(92, 153, 92, 203, 52, 178, ILI9341_BLACK);  //  Dibuja sombra trian izq
  tft.fillRect(90, 160, 40, 30, ILI9341_CYAN);  // Define rectángulo lleno izq.
  tft.fillTriangle(90, 150, 90, 200, 50, 175, ILI9341_CYAN);  // Define triángulo lleno izq.

  tft.fillRect(192, 163, 42, 33, ILI9341_BLACK);  //  Dibuja sombra
  tft.fillTriangle(232, 153, 232, 203, 272, 178, ILI9341_BLACK);  //  Dibuja sombra
  tft.fillRect(190, 160, 40, 30, ILI9341_RED);  // Define rectángulo lleno der.
  tft.fillTriangle(230, 150, 230, 200, 270, 175, ILI9341_RED);  // Define triángulo lleno der.

}
```

```
void loop(void) {

  // See if there's any  touch data for us
  if (!ts.bufferEmpty())
  {
    // Retrieve a point
    TS_Point p = ts.getPoint();
    // Scale using the calibration #'s
    // and rotate coordinate system
    p.x = map(p.x, TS_MINY, TS_MAXY, 0, tft.height());
    p.y = map(p.y, TS_MINX, TS_MAXX, 0, tft.width());
    int y = tft.height() - p.x;
    int x = p.y;

    Serial.print("Btn hit");
    Serial.print(x);
    Serial.print("\t");
    Serial.print(y);
    Serial.print("\t");     // Envía las coordenadas en que fue presionada la pantalla
    Serial.println(valor);  // Envía el contenido de la variable valor

    if((x > REDBUTTON_X1) && (x < (REDBUTTON_X2))) {
        if ((y > REDBUTTON_Y1) && (y <= (REDBUTTON_Y2))) {
            tft.setCursor(170, 60);
            tft.setTextColor(ILI9341_DARKGREY);
            tft.print(valor);
            tft.setTextColor(ILI9341_CYAN);
            tft.fillRect(190, 160, 40, 30, ILI9341_DARKGREY);  // Se borra el botón
            tft.fillTriangle(230, 150, 230, 200, 270, 175, ILI9341_DARKGREY);  // Se borra el botón
            tft.fillRect(192, 163, 42, 33, ILI9341_RED);  //  La sombra se convierte en botón
            tft.fillTriangle(232, 153, 232, 203, 272, 178, ILI9341_RED);  //  La sombra se convierte en botón
           valor = valor + 1;
           Serial.println("Red btn hit");
           delay(100);
           while(!ts.bufferEmpty()){
             TS_Point p = ts.getPoint();   // Vacía el búffer
           }
           tft.setCursor(10, 60);
  tft.print("Valor = ");

  tft.setCursor(170, 60);
  tft.println(valor);
// Vuelve a dibujar el botón con sombra
  tft.fillRect(192, 163, 42, 33, ILI9341_BLACK);  //  Dibuja sombra
  tft.fillTriangle(232, 153, 232, 203, 272, 178, ILI9341_BLACK);  //  Dibuja sombra
  tft.fillRect(190, 160, 40, 30, ILI9341_RED);  // Define rectángulo lleno der.
```

```
  tft.fillTriangle(230, 150, 230, 200, 270, 175, ILI9341_RED);  // Define triángulo lleno der.
        }
       }

    if((x > CYANBUTTON_X1) && (x < (CYANBUTTON_X2))) {
       if ((y > CYANBUTTON_Y1) && (y <= (CYANBUTTON_Y2))) {
           tft.setCursor(170, 60);
           tft.setTextColor(ILI9341_DARKGREY);
           tft.print(valor);
           tft.setTextColor(ILI9341_CYAN);
           tft.fillRect(90, 160, 40, 30, ILI9341_DARKGREY);  // Se borra el botón
           tft.fillTriangle(90, 150, 90, 200, 50, 175, ILI9341_DARKGREY);  // Se borra el botón
           tft.fillRect(92, 163, 42, 33, ILI9341_CYAN);  //  La sombra se vuelve el botón
           tft.fillTriangle(92, 153, 92, 203, 52, 178, ILI9341_CYAN);  // La sombra se vuelve el botón
         valor = valor -1;
         Serial.println("Cyan btn hit");
         delay(100);
          while(!ts.bufferEmpty()){
            TS_Point p = ts.getPoint();   // Vacía el búffer
           }
           tft.setCursor(10, 60);
  tft.print("Valor = ");
  tft.setCursor(170, 60);
    tft.print("    ");
  tft.setCursor(170, 60);
  tft.println(valor);
 // Vuelve a dibujar el botón con sombra
  tft.fillRect(92, 163, 42, 33, ILI9341_BLACK);  //  Dibuja sombra rec izq
  tft.fillTriangle(92, 153, 92, 203, 52, 178, ILI9341_BLACK);  //  Dibuja sombra trian izq
  tft.fillRect(90, 160, 40, 30, ILI9341_CYAN);  // Define rectángulo lleno izq.
  tft.fillTriangle(90, 150, 90, 200, 50, 175, ILI9341_CYAN);  // Define triángulo lleno izq.
       }
      }
  }
}
```

Anexo B. Tabla de definición de colores

La tabla de colores predefinidos (en formato RGB), que fue tomada de las bibliotecas, se reproduce a continuación.

```
// Color definitions
#define ILI9341_BLACK       0x0000      /*   0,   0,   0 */
#define ILI9341_NAVY        0x000F      /*   0,   0, 128 */
#define ILI9341_DARKGREEN   0x03E0      /*   0, 128,   0 */
#define ILI9341_DARKCYAN    0x03EF      /*   0, 128, 128 */
#define ILI9341_MAROON      0x7800      /* 128,   0,   0 */
#define ILI9341_PURPLE      0x780F      /* 128,   0, 128 */
#define ILI9341_OLIVE       0x7BE0      /* 128, 128,   0 */
#define ILI9341_LIGHTGREY   0xC618      /* 192, 192, 192 */
#define ILI9341_DARKGREY    0x7BEF      /* 128, 128, 128 */
#define ILI9341_BLUE        0x001F      /*   0,   0, 255 */
#define ILI9341_GREEN       0x07E0      /*   0, 255,   0 */
#define ILI9341_CYAN        0x07FF      /*   0, 255, 255 */
#define ILI9341_RED         0xF800      /* 255,   0,   0 */
#define ILI9341_MAGENTA     0xF81F      /* 255,   0, 255 */
#define ILI9341_YELLOW      0xFFE0      /* 255, 255,   0 */
#define ILI9341_WHITE       0xFFFF      /* 255, 255, 255 */
#define ILI9341_ORANGE      0xFD20      /* 255, 165,   0 */
#define ILI9341_GREENYELLOW 0xAFE5      /* 173, 255,  47 */
#define ILI9341_PINK        0xF81F
```

Referencias Bibliográficas

Atmel. (2014) ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet. USA. Atmel Corporation.

Burgess, P. (2014) Adafruit GFX Graphics Library. USA. Adafruit Industries.

Burgess, P. (2016). Adafruit GFX Graphics Library Overview. Adafruit Industries. Disponible desde: <https://learn.adafruit.com/adafruit-gfx-graphics-library/overview> [Acceso el 7 de febrero 2016].

Downs, R. (2005) Using resistive touch screens for human/machine interface**. Analog Applications Journal**, 3Q 2005. p. 5-9.

Walker, G. (2012) A review of technologies for sensing contact location on the surface of a display. **Journal of the Society for Information Display**, 20(8), p. 413-440.