```
/*
Author: Danny van den Brande, ArduinoSensors.nl. Bluecore Tech.
 */
#include <SPFD5408_Adafruit_GFX.h>    // Core graphics Adafruit library
#include <SPFD5408_Adafruit_TFTLCD.h> // Hardware-specific library
#include <SPFD5408_TouchScreen.h>     // Touchscreen library

// *** Define Touchscreen Pin
#define YP A3
#define XM A2
#define YM 9
#define XP 8
// *** Define Value of Touchscreen input
#define TS_MINX 150
#define TS_MINY 120
#define TS_MAXX 920
#define TS_MAXY 940

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

// *** Define Pin of LCD used
#define LCD_CS A3
#define LCD_CD A2
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4

// *** Define Name of Color
#define BLACK   0x0000
#define WHITE   0xFFFF
#define RED     0xF800
#define GREEN   0x07E0
#define BLUE    0x001F
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define GREY    0x2108
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
#define MINPRESSURE 10
#define MAXPRESSURE 1000

// Meter colour schemes
#define RED2RED 0
#define GREEN2GREEN 1
#define BLUE2BLUE 2
#define BLUE2RED 3
#define GREEN2RED 4
#define RED2GREEN 5

uint32_t runTime = -99999;       // time for next update
int reading = 0; // Value to be displayed
int d = 0; // Variable used for the sinewave test waveform
```

```
boolean alert = 0;
int8_t ramp = 1;
int tesmod =0;
void setup() {

tft.reset(); //Reset LCD to begin

tft.begin(0x9341); //SDFP5408 Chipset ID on Library

tft.setRotation(3); // Set Rotation at 0 degress (default)

tft.fillScreen(BLACK); //Set Background Color with BLACK
   tft.setCursor (7,208);
   tft.setTextSize (3);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("HUMIDITY");
   tft.setCursor (75,78);
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("TEMPERATURE");
   tft.setCursor (70,66);
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("BLUECORE TECH");

   tft.setCursor (242,20);
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("FAHRENHEIT");

   tft.setCursor (242,118);//heat index fahrenheit
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("HEAT INDEX");

   tft.setCursor (238,205);//heat index fahrenheit
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("HEAT INDEX C");

   tft.setCursor (265,220);//Small celcius text
   tft.setTextSize (1);
   tft.setTextColor (WHITE,BLACK);
   tft.print ("CELSIUS");

//Design Interface (lines)
   tft.fillRect(0,197,320,4,BLUE);
   tft.fillRect(0,236,320,4,BLUE);
   tft.fillRect(217,0,4,240,BLUE);
   tft.fillRect(217,98,320,4,BLUE);

}
```

```
void loop() {
  float h = 90;
  float t = 90;
  float f = 90;
  // Compute heat index in Fahrenheit (the default)
  float hif = 90;
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = 90;

if (millis() - runTime >= 500) { // Execute every 500ms
    runTime = millis();
    if(tesmod==0){reading=99; }
    //if(tesmod==1){reading = t;}

    int xpos = 0, ypos = 5, gap = 4, radius = 52;
    // Draw a large meter
    xpos = 320/2 - 160, ypos = 0, gap = 100, radius = 105;
    ringMeter(reading,0,100, xpos,ypos,radius,"Celsius",GREEN2RED); // Draw analogue meter

  if(h>0){  //Humidity %
    tft.setCursor (157,208);
    tft.setTextSize (3);
    tft.setTextColor (BLUE,BLACK);
    tft.print (h,0); tft.print ('%');
    tesmod=1;
   }
  if(f>0){  //Fahrenheit
    tft.setCursor (237,38);
    tft.setTextSize (4);
    tft.setTextColor (BLUE,BLACK);
    tft.print (f,0);
    tesmod=1;
   }
  if(hif>0){  //Heat index Fahrenheit
    tft.setCursor (237,138);
    tft.setTextSize (4);
    tft.setTextColor (BLUE,BLACK);
    tft.print (hif,0);
    tesmod=1;
   }
  if(hic>0){  //Heat index Celsius
    tft.setCursor (238,217);
    tft.setTextSize (2);
    tft.setTextColor (BLUE,BLACK);
    tft.print (hic,0);
    tesmod=1;
   }
   }
}
```

```
// #######################################################################
//  Draw the meter on the screen, returns x coord of righthand side
// #######################################################################
int ringMeter(int value, int vmin, int vmax, int x, int y, int r, char *units, byte scheme)
{
  // Minimum value of r is about 52 before value text intrudes on ring
  // drawing the text first is an option

  x += r; y += r;   // Calculate coords of centre of ring
  int w = r / 3;    // Width of outer ring is 1/4 of radius
  int angle = 150;  // Half the sweep angle of meter (300 degrees)
  int v = map(value, vmin, vmax, -angle, angle); // Map the value to an angle v
  byte seg = 3; // Segments are 3 degrees wide = 100 segments for 300 degrees
  byte inc = 6; // Draw segments every 3 degrees, increase to 6 for segmented ring
  // Variable to save "value" text colour from scheme and set default
  int colour = BLUE;

  // Draw colour blocks every inc degrees
  for (int i = -angle+inc/2; i < angle-inc/2; i += inc) {
    // Calculate pair of coordinates for segment start
    float sx = cos((i - 90) * 0.0174532925);
    float sy = sin((i - 90) * 0.0174532925);
    uint16_t x0 = sx * (r - w) + x;
    uint16_t y0 = sy * (r - w) + y;
    uint16_t x1 = sx * r + x;
    uint16_t y1 = sy * r + y;

    // Calculate pair of coordinates for segment end
    float sx2 = cos((i + seg - 90) * 0.0174532925);
    float sy2 = sin((i + seg - 90) * 0.0174532925);
    int x2 = sx2 * (r - w) + x;
    int y2 = sy2 * (r - w) + y;
    int x3 = sx2 * r + x;
    int y3 = sy2 * r + y;

    if (i < v) { // Fill in coloured segments with 2 triangles
      switch (scheme) {
        case 0: colour = RED; break; // Fixed colour
        case 1: colour = GREEN; break; // Fixed colour
        case 2: colour = BLUE; break; // Fixed colour
        case 3: colour = rainbow(map(i, -angle, angle, 0, 127)); break; // Full spectrum blue to red
        case 4: colour = rainbow(map(i, -angle, angle, 70, 127)); break; // Green to red (high
temperature etc)
        case 5: colour = rainbow(map(i, -angle, angle, 127, 63)); break; // Red to green (low battery
etc)
        default: colour = BLUE; break; // Fixed colour
       }
      tft.fillTriangle(x0, y0, x1, y1, x2, y2, colour);
      tft.fillTriangle(x1, y1, x2, y2, x3, y3, colour);
      //text_colour = colour; // Save the last colour drawn
    }
    else // Fill in blank segments
```

```
    {
      tft.fillTriangle(x0, y0, x1, y1, x2, y2, GREY);
      tft.fillTriangle(x1, y1, x2, y2, x3, y3, GREY);
    }
  }
  // Convert value to a string
  char buf[10];
  byte len = 2; if (value > 999) len = 4;
  dtostrf(value, len, 0, buf);
  buf[len] = ' '; buf[len] = 0; // Add blanking space and terminator, helps to centre text too!
  // Set the text colour to default
  tft.setTextSize(1);

  if(value>9){
  tft.setTextColor(colour,BLACK);
  tft.setCursor(x-25,y-10);tft.setTextSize(5);
  tft.print(buf);}
  if(value<10){
  tft.setTextColor(colour,BLACK);
  tft.setCursor(x-25,y-10);tft.setTextSize(5);
  tft.print(buf);}

  tft.setTextColor(WHITE,BLACK);

  tft.setCursor(x-39,y+75);tft.setTextSize(2);
  tft.print(units); // Units display

  // Calculate and return right hand side x coordinate
  return x + r;
}

// ##########################################################################
// Return a 16 bit rainbow colour
// ##########################################################################
unsigned int rainbow(byte value)
{
  // Value is expected to be in range 0-127
  // The value is converted to a spectrum colour from 0 = blue through to 127 = red

  byte red = 0; // Red is the top 5 bits of a 16 bit colour value
  byte green = 0;// Green is the middle 6 bits
  byte blue = 0; // Blue is the bottom 5 bits
  byte quadrant = value / 32;

  if (quadrant == 0) {
    blue = 31;
    green = 2 * (value % 32);
    red = 0;
  }
  if (quadrant == 1) {
    blue = 31 - (value % 32);
    green = 63;
```

```
    red = 0;
  }
  if (quadrant == 2) {
    blue = 0;
    green = 63;
    red = value % 32;
  }
  if (quadrant == 3) {
    blue = 0;
    green = 63 - 2 * (value % 32);
    red = 31;
  }
  return (red << 11) + (green << 5) + blue;
}

// ###########################################################################
// Return a value in range -1 to +1 for a given phase angle in degrees
// ###########################################################################
float sineWave(int phase) {
  return sin(phase * 0.0174532925);
}
```