```
/*
 * This is a program that monitors a DHT sensor for Temperature and Humidity
 * and feeds a PI Controller that has adjustable setpoints, gains, and humidity
 * alarm setpoint that also works with many different sensors (just uncomment out
 * sensor type and change the data pin number if not using the same one "D1")
 *
 * The Program Offers the following Functions:
 * ------------------------------------------
 * 1) Displays the Temperature and Humidty Locally on Display (analog & digital values)
 * 2) Contains on-Screen Instructions for setting Up WiFi. (Future)
 * 3) Diplays common error messages for Server and Wifi. (Future)
 * 4) Detects Sensor failure and displays an error message.
 * 5) Top Left Buttun Resets WiFi Settings. (Future)
 * 6) The Web-Server operates by typing in the Wio Device IP
 *    Address in a Browser Window and Displays the current
 *    Temperature and Humidy with an "Auto-Refresh" of Browser
 *    Every 5 seconds.
 * 7) Contains a Built in WiFi Manager to connect Wio Device
 *    to your home router via a Graphical User Interface. (Future)
 * 8) Provides the following button functionality:
 *    - Bottom Right swith (Push in) = Menu Operations to set parameters
 *    - Top Right Button = (+) to adjust menu parameters
 *    - Top Middle Button = (-) to adjust menu parameters
 *    - Top Left Button = (Enter) to store menu parameters
 * 9) Fan Control is PWM controlled through pin "A8"
 * 10) Parameters are stored in EEPROM and read into program once started.
 * 11) Contains an internal buzzer when humidity falls below the user setpoint.
 */

// Libraries:
// ----------
#include <FlashStorage_SAMD.h>
#include <TFT_eSPI.h> // Hardware-specific library
#include <SPI.h>
#include "DHT.h" // Groove DHT Temperature $ Humidity library
TFT_eSPI tft = TFT_eSPI();       // Invoke custom library

// Global Variables:
// -----------------
#define TFT_GREY 0x5AEB
int count = 0; // Menu Counter
int Kp = 50;  // Proportional Gain (must be less than 255)
int Ki = 5;  // Integral Gain (must be less than 255)
int address = 100;
int flag = 0; // Program Flag to lock menu
int Ha = 10; // Humidity Alarm Setpoint
int Sp = 70; // Controller Setpoint
int PI_Out; // Custom Control Function Return Value
const double delta_time = 1.2; // 0.5 Second Sample Rate in Auto (glaobal variable)
double I_Term = 0.0; // Integral Term (global variable)
double output = 0.0;
const double windup_guard = 60.0; // Integral Windup prevention
double error = 0.0;
double Hum; // Humidity storage Variable
double TemperatureC; // Temperature storage variable for Deg C
double TempF; // Temperature storage variable for Deg F
int Fs; // % Fan Speed
// double h = 68; // Test Value, replace with actual humidity reading
unsigned long startMillis;  // Non Latency Timed Function
unsigned long currentMillis;
const unsigned long period = 1000;  //the value is a number of milliseconds (3 seconds)
unsigned long startMillis1;  // Non Latency Timed Function
unsigned long currentMillis1;
const unsigned long period1 = 6000;  //the value is a number of milliseconds (6 seconds)
#define FLASH_DEBUG  0
#define TFT_GREY 0x5AEB
#define LOOP_PERIOD 35 // Display updates every 35 ms
float ltx = 0;    // Saved x coord of bottom of needle
uint16_t osx = 120, osy = 120; // Saved x & y coords (osx = 120, osy = 120)
```

```
uint32_t updateTime = 0;       // time for next update
int old_analog =  -999; // Value last displayed
int old_digital = -999; // Value last displayed
int value[6] = {0, 0, 0, 0, 0, 0};
int old_value[6] = { -1, -1, -1, -1, -1, -1};
int d = 0;
boolean interlock = true; // Stops Program execution while in Menu

// DHT Sensor Characteristics (Uncomment whatever type you're using)
// ----------------------------------------------------------------
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
//#define DHTTYPE DHT10   // DHT 10
//#define DHTTYPE DHT20   // DHT 20
#define DHTPIN D1      // Data Pin we're connected to
DHT dht(DHTPIN, DHTTYPE);   //   DHT11 DHT21 DHT22
//DHT dht(DHTTYPE);         //   DHT10 DHT20 don't need to define Pin

// Motor Drive Pin:
// ----------------
#define PWM_Pin A8 // Motor Drive Pin

// Main Program:
// =============

void setup() {
  Serial.begin(115200);
  tft.init();
  tft.setRotation(3);
  //tft.setTextSize(2);
  tft.fillScreen(TFT_BLACK);
  tft.setTextColor(TFT_WHITE);
  tft.drawString("Cigar Humidor Controller", 10, 10, 4); //prints strings from (x, y,
  font size)
  tft.drawString("With Advanced Features", 10, 50, 4);
  tft.drawString("By; Roy H Guerra Jr.", 10, 90, 4);
  pinMode(WIO_5S_UP, INPUT_PULLUP);  // Enable Wio Button puulup Resistors
  pinMode(WIO_5S_DOWN, INPUT_PULLUP);
  pinMode(WIO_5S_LEFT, INPUT_PULLUP);
  pinMode(WIO_5S_RIGHT, INPUT_PULLUP);
  pinMode(WIO_5S_PRESS, INPUT_PULLUP);
  pinMode(WIO_KEY_A, INPUT_PULLUP);
  pinMode(WIO_KEY_B, INPUT_PULLUP);
  pinMode(WIO_KEY_C, INPUT_PULLUP);
  pinMode(PWM_Pin, OUTPUT); // PWM Channel
  pinMode(WIO_BUZZER, OUTPUT); // Internal Wio Buzzer
  dht.begin(); // Initialize DHT sensor
  delay(2000);  // 2S loop delay
  tft.fillScreen(TFT_BLACK);
  updateTime = millis(); // Next update time
  startMillis = millis();  //initial time stamp
  startMillis1 = millis();  //initial time stamp
  analogMeter(); // Draw analog meter
  plotLinear("oF", 260, 70); // Draw 1 linear meters
}

void loop() {
   currentMillis = millis();  // Get a time Stamp
   currentMillis1 = millis(); // Get a time Stamp
   if (digitalRead(WIO_5S_PRESS) == LOW) {
     Serial.println("5 Way Button Press");
     interlock = false; // Set interlock
     count = 1; // Set Counter
     Serial.println("Count = " + String(count));
    }
   switch (count) {
    case 1:
     tft.fillScreen(TFT_BLACK);
```

```
        tft.setTextColor(TFT_CYAN);
        tft.drawString("Set Humidity Alarm SP", 10, 10, 4); //prints strings from (x, y, font size)
        tft.drawString("-------------------------------", 10, 30, 4);
        tft.setTextColor(TFT_YELLOW);
        tft.drawString("Press Top Right Button (+)", 10, 70, 4);
        tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
        tft.setTextColor(TFT_WHITE);
        tft.drawString("Humidity Alarm SP = ", 10, 160, 4);
        tft.drawRect(245,150,55,35,TFT_WHITE);
        tft.drawString(String(Ha), 250, 160, 4);
        tft.setTextColor(TFT_RED);
        tft.drawString("Press Top Left Button To", 10, 192, 4);
        tft.drawString("Save Configuration (exit)", 10, 215, 4);
        flag = 1; // Change program flag
        while (flag == 1) {
          if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 1)){
            Serial.println("B Key pressed");
            if (Ha > 0){
               Ha -= 1;
               tft.fillRect(245,150,55,35,TFT_BLACK);
               tft.drawRect(245,150,55,35,TFT_WHITE);
               tft.setTextColor(TFT_WHITE);
               tft.drawString(String(Ha), 250, 160, 4);
              }
            Serial.println("Ha = " + String(Ha));
            }
            if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 1)) {
               Serial.println("A Key pressed");
               if (Ha < 80){
                 Ha += 1;
                 tft.fillRect(245,150,55,35,TFT_BLACK);
                 tft.drawRect(245,150,55,35,TFT_WHITE);
                 tft.setTextColor(TFT_WHITE);
                 tft.drawString(String(Ha), 250, 160, 4);
                }
             Serial.println("Ha = " + String(Ha));
             }
            if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 1)) {
              Serial.println("C Key pressed");
              /***
              The function EEPROM.update(address, val) is equivalent to the following:
              if( EEPROM.read(address) != val ) {
               EEPROM.write(address, val);
              }
               ***/
               EEPROM.update(address + 20, Ha);
               EEPROM.commit();
               Serial.println("Wrote " + String(Ha) + " To EEPROM  Address 120");
               tft.fillScreen(TFT_BLACK);
               tft.setTextColor(TFT_CYAN);
               tft.drawString("Humidity Alarm SP Saved", 10, 70, 4);
               delay(2000);  // 2S loop delay
               tft.fillScreen(TFT_BLACK);
               flag = 0;  // Reset flag
               count = 2; // Next menu Option
             }
        delay(150);  // 150mS loop delay
        }
        break;
       case 2:
        tft.fillScreen(TFT_BLACK);
        tft.setTextColor(TFT_MAGENTA);
        tft.drawString("Set Controller SP", 10, 10, 4); //prints strings from (x, y, font size)
        tft.drawString("------------------------", 10, 30, 4);
        tft.setTextColor(TFT_YELLOW);
        tft.drawString("Press Top Right Button (+)", 10, 70, 4);
        tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
        tft.setTextColor(TFT_WHITE);
        tft.drawString("    Controller SP = ", 10, 160, 4);
        tft.drawRect(245,150,55,35,TFT_WHITE);
        tft.drawString(String(Sp), 250, 160, 4);
        tft.setTextColor(TFT_RED);
        tft.drawString("Press Top Left Button To", 10, 192, 4);
        tft.drawString("Save Configuation (exit)", 10, 215, 4);
        flag = 1; // Change program flag
        while (flag == 1) {
          if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 2)){
            Serial.println("B Key pressed");
            if (Sp > 50){
               Sp -= 1;
               tft.fillRect(245,150,55,35,TFT_BLACK);
               tft.drawRect(245,150,55,35,TFT_WHITE);
               tft.setTextColor(TFT_WHITE);
               tft.drawString(String(Sp), 250, 160, 4);
              }
            Serial.println("Ha = " + String(Sp));
            }
            if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 2)) {
               Serial.println("A Key pressed");
               if (Sp < 80){
                 Sp += 1;
                 tft.fillRect(245,150,55,35,TFT_BLACK);
                 tft.drawRect(245,150,55,35,TFT_WHITE);
                 tft.setTextColor(TFT_WHITE);
                 tft.drawString(String(Sp), 250, 160, 4);
                }
             Serial.println("Ha = " + String(Sp));
             }
            if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 2)) {
              Serial.println("C Key pressed");
              /***
              The function EEPROM.update(address, val) is equivalent to the following:
              if( EEPROM.read(address) != val ) {
                EEPROM.write(address, val);
              }
               ***/
               EEPROM.update(address + 30, Sp);
               EEPROM.commit();
               Serial.println("Wrote " + String(Sp) + " To EEPROM  Address 130");
               tft.fillScreen(TFT_BLACK);
               tft.setTextColor(TFT_MAGENTA);
               tft.drawString("Controller Setpoint Saved", 10, 70, 4);
               delay(2000);  // 2S loop delay
               tft.fillScreen(TFT_BLACK);
               flag = 0;  // Reset flag
               count = 3; // Next menu Option
             }
        delay(150);  // 150mS loop delay
        }
        break;
      case 3:
        tft.fillScreen(TFT_BLACK);
        tft.setTextColor(TFT_GREEN);
        tft.drawString("Set Proportional Gain", 10, 10, 4); //prints strings from (x, y,
        font size)
        tft.drawString("------------------------------", 10, 30, 4);
        tft.setTextColor(TFT_YELLOW);
        tft.drawString("Press Top Right Button (+)", 10, 70, 4);
        tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
        tft.setTextColor(TFT_WHITE);
        tft.drawString("Proportional Gain = ", 10, 160, 4);
        tft.drawRect(245,150,55,35,TFT_WHITE);
        tft.drawString(String(Kp), 250, 160, 4);
        tft.setTextColor(TFT_RED);
        tft.drawString("Press Top Left Button To", 10, 192, 4);
        tft.drawString("Save Configuration (exit)", 10, 215, 4);
        flag = 1; // Change program flag
        while (flag == 1) {
          if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 3)){
            Serial.println("B Key pressed");
            if (Kp > 1){
               Kp -= 1;
               tft.fillRect(245,150,55,35,TFT_BLACK);
               tft.drawRect(245,150,55,35,TFT_WHITE);
               tft.setTextColor(TFT_WHITE);
               tft.drawString(String(Kp), 250, 160, 4);
              }
            Serial.println("Kp = " + String(Kp));
            }
            if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 3)) {
               Serial.println("A Key pressed");
               if (Kp < 100){
                 Kp += 1;
                 tft.fillRect(245,150,55,35,TFT_BLACK);
                 tft.drawRect(245,150,55,35,TFT_WHITE);
                 tft.setTextColor(TFT_WHITE);
                 tft.drawString(String(Kp), 250, 160, 4);
                }
             Serial.println("Kp = " + String(Kp));
             }
            if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 3)) {
              Serial.println("C Key pressed");
              /***
              The function EEPROM.update(address, val) is equivalent to the following:
              if( EEPROM.read(address) != val ) {
               EEPROM.write(address, val);
              }
               ***/
               EEPROM.update(address, Kp);
               EEPROM.commit();
               Serial.println("Wrote " + String(Kp) + " To EEPROM  Address 100");
               tft.fillScreen(TFT_BLACK);
               tft.setTextColor(TFT_GREEN);
               tft.drawString("Proportiional Gain Saved", 10, 70, 4);
               delay(2000);  // 2S loop delay
               tft.fillScreen(TFT_BLACK);
               flag = 0;  // Reset flag
               count = 4; // Next menu Option
             }
        delay(150);  // 150mS loop delay
        }
        break;
       case 4:
        tft.fillScreen(TFT_BLACK);
        tft.setTextColor(TFT_BLUE);
        tft.drawString("Set Integral Gain", 10, 10, 4); //prints strings from (x, y, font
        size)
        tft.drawString("------------------------", 10, 30, 4);
        tft.setTextColor(TFT_YELLOW);
        tft.drawString("Press Top Right Button (+)", 10, 70, 4);
        tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
        tft.setTextColor(TFT_WHITE);
        tft.drawString("    Integral Gain = ", 10, 160, 4);
        tft.drawRect(245,150,55,35,TFT_WHITE);
        tft.drawString(String(Ki), 250, 160, 4);
        tft.setTextColor(TFT_RED);
        tft.drawString("Press Top Left Button To", 10, 192, 4);
        tft.drawString("Save Configuration (exit)", 10, 215, 4);
        flag = 1; // Change program flag
        while (flag == 1) {
          if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 4)){
            Serial.println("B Key pressed");
            if (Ki > 0){
               Ki -= 1;
               tft.fillRect(245,150,55,35,TFT_BLACK);
               tft.drawRect(245,150,55,35,TFT_WHITE);
               tft.setTextColor(TFT_WHITE);
               tft.drawString(String(Ki), 250, 160, 4);
              }
            Serial.println("Ki = " + String(Ki));
            }
            if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 4)) {
               Serial.println("A Key pressed");
               if (Ki < 50){
                 Ki += 1;
                 tft.fillRect(245,150,55,35,TFT_BLACK);
                 tft.drawRect(245,150,55,35,TFT_WHITE);
                 tft.setTextColor(TFT_WHITE);
                 tft.drawString(String(Ki), 250, 160, 4);
                }
             Serial.println("Ki = " + String(Ki));
             }
            if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 4)) {
              Serial.println("C Key pressed");
              /***
              The function EEPROM.update(address, val) is equivalent to the following:
              if( EEPROM.read(address) != val ) {
               EEPROM.write(address, val);
              }
               ***/
               EEPROM.update(address + 10, Ki);
               EEPROM.commit();
               Serial.println("Wrote " + String(Ki) + " To EEPROM  Address 110");
               tft.fillScreen(TFT_BLACK);
               tft.setTextColor(TFT_BLUE);
               tft.drawString("Integral Gain Saved", 10, 70, 4);
               delay(2000);  // 2S loop delay
               tft.fillScreen(TFT_BLACK);
               flag = 0;  // Reset flag
               interlock = true; // Reset Interlock
               count = 0; // Next menu Option
               NVIC_SystemReset(); // Re-Start Program
             }
        delay(150);  // 150mS loop delay
        }
       break;
       default:
        count = 0; // Default
       break;
      }
    // delay(100);  // 100mS loop delay
    if (interlock == true){
     // No Operation of Program past this point once interlock is set while in menu's
    if (currentMillis - startMillis >= period){  // Test whether the period has elapsed
        SensorData(); // Goto Function
        controller(); // Goto Function
        // Plot Fan Drive
        tft.setTextColor(TFT_MAGENTA);  // Text color
        tft.drawString("Fan Drive = ", 55, 58, 2); // SCREEN Header
        tft.fillRect(135,56,48,20,TFT_BLACK);  // Draw a Rect to erase previous data
        tft.drawRect(135,56,48,20,TFT_MAGENTA);  // Draw a Rect.
        tft.drawString(String(Fs) + " %", 140, 58, 2); //prints strings from (x, y)
        startMillis = currentMillis;  // New Time Stamp
       }
    if (currentMillis1 - startMillis1 >= period1){  // Test whether the period has elapsed
        buzzer(); // Goto Function
        startMillis1 = currentMillis1;  // New Time Stamp
      }
     if (updateTime <= millis()) {
        updateTime = millis() + LOOP_PERIOD;
      //value[0] = map(analogRead(A0), 0, 1023, 0, 100); // Test with an actual value
      // value[0] = 50 + 50 * sin((d + 0) * 0.0174532925);   // Create a Sine wave for
      testing
       plotPointer(); // Goto Function
       plotNeedle(int(Hum), 0);  // Goto Function
      }
    }
   }

   // PI Controller Function:
   // -----------------------
   float Calculate_PI () {
     // Read EEPROM Kp & Ki, Ha, & Sp:
      Kp = EEPROM.read(address);
      Serial.println("Kp = " + String(Kp));
      Ki = EEPROM.read(address + 10);
      Serial.println("Ki = " + String(Ki));
      Ha = EEPROM.read(address + 20);
      Serial.println("Ha = " + String(Ha));
      Sp = EEPROM.read(address + 30);
      Serial.println("Sp = " + String(Sp));
      if ((Kp == 255) || (Ki == 255) || (Sp == 255) || (Ha ==255)) { // Guards against EEPROM not being set
        Kp = 50;
        Ki = 5;
        Ha = 50;
        Sp = 70;
      }
      error = Sp - Hum; // Error Term, h = feedback
     I_Term += (error * delta_time); // Intergral Term
     if (I_Term > windup_guard){ // Positive Integral Windup Guard
       I_Term = windup_guard;
     }
     if (I_Term < - windup_guard){ // Negative Integral Windup Guard
       I_Term = - windup_guard;
     }
     if (isnan(I_Term)){ // Reset if NAN
       I_Term = 0;
     }
    output = (Kp * error) + (Ki * I_Term); // Controller Output (Proportional + Integral)
    output = constrain(output, 0, 255); // Limits Controller Range
    Serial.println("Kp = " + String(Kp)); // Debug
    Serial.println("Ki = " + String(Ki)); // Debug
    Serial.println("Setpoint = " + String(Sp)); // Debug
    Serial.println("Feedback (humidity) = " + String(Hum)); // Debug
    Serial.println("Error = " + String(error)); // Debug
    Serial.println("I_Term = " + String(I_Term)); // Debug
    Serial.println("Ki *I_Term = " + String(Ki * I_Term)); // Debug
    Serial.println("P_Term = " + String(Kp * error)); // Debug
    Serial.println("Output = " + String(output)); // Debug
    Serial.println("Alarm Setpoint = " + String(Ha)); // Debug
    return int(output); // Return PI Control Value as an integer
   }

   // Function to Sound Buzzer:
   // -------------------------
   void buzzer(){ // Buzzer Function Block
     if (Hum < Ha){
       analogWrite(WIO_BUZZER, 128);
       delay(1000);
       analogWrite(WIO_BUZZER, 0);
       delay(1000);
      }
   }

   // Function to Read Control Loop and set PWM and Speed Indication:
   // ---------------------------------------------------------------
   void controller(){ // Controller Function Block
     PI_Out = Calculate_PI(); // Calculate new PI Control Value
     Serial.println("PI_Out = " + String(PI_Out)); // Debug
     analogWrite(PWM_Pin, PI_Out); // PWM Value (0-255)
     Fs = map(output, 0, 255, 0, 100); // Rescale controller output to % fan speed
     Serial.println("Fan Speed = " + String(Fs)); // Debug
```

```
}

// Read Sensor Function
// --------------------
void SensorData(){
  Hum = dht.readHumidity(); // Measure the humidity
  Serial.println("Humidity = " + String(Hum));
  TemperatureC = dht.readTemperature(); // Measure the temperature
  TempF = ((TemperatureC * 9/5) + 32); // Convert temperature to degrees Fahrenheit
  Serial.println("Temperature = " + String(TempF));
  // Compare temperature & humidity events and perform a check sum.
  if (isnan(TemperatureC) || isnan(Hum)){ // Print "0" for a bad reading
    TempF = 0;
    Hum = 0;
    Serial.println("Bad Connection or Sensor");
   }
}

// Draw the Horizontal Analog Meter & Menu on the screen
// -----------------------------------------------------
void analogMeter() {
    // Meter outline
    tft.fillRect(0, 85, 239, 126, TFT_GREY); // 0, 0, 239, 126 (x, y, w, h)
    tft.fillRect(5, 88, 230, 119, TFT_WHITE); // 5, 3, 230, 119,
    //tft.fillRect(5, 10, 100, 50, TFT_WHITE); // SCREEN Header
    tft.setTextColor(TFT_WHITE);
    tft.drawString(" Cigar Humidor Parameters", 5, 10, 4); // SCREEN Header
    tft.drawString(" -------------------------------------", 5, 35, 4); // SCREEN Header
    tft.setTextColor(TFT_BLACK);  // Text color
    // Draw ticks every 5 degrees from -50 to +50 degrees (100 deg. FSD swing)
    for (int i = -50; i < 51; i += 5) {
        // Long scale tick length
        int tl = 15;
        // Coodinates of tick to draw
        float sx = cos((i - 90) * 0.0174532925);
        float sy = sin((i - 90) * 0.0174532925);
        uint16_t x0 = sx * (100 + tl) + 120; // 120
        uint16_t y0 = sy * (100 + tl) + 220; // 140
        uint16_t x1 = sx * 100 + 120; // 120
        uint16_t y1 = sy * 100 + 220; // 140
        // Coordinates of next tick for zone fill
        float sx2 = cos((i + 5 - 90) * 0.0174532925);
        float sy2 = sin((i + 5 - 90) * 0.0174532925);
        int x2 = sx2 * (100 + tl) + 120; // 120
        int y2 = sy2 * (100 + tl) + 220; // 140
        int x3 = sx2 * 100 + 120; // 120
        int y3 = sy2 * 100 + 220; // 140
        // Yellow zone limits
        if (i >= -50 && i < 1) {
          tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_YELLOW);
          tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_YELLOW);
        }
        // Green zone limits
        if (i >= 1 && i < 25) {  // 0
            tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_GREEN);
            tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_GREEN);
        }
        // Orange zone limits
        if (i >= 25 && i < 50) {
           tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_ORANGE);
           tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_ORANGE);
        }
        // Short scale tick length
        if (i % 25 != 0) {
            tl = 8;
        }
        // Recalculate coords incase tick lenght changed
        x0 = sx * (100 + tl) + 120; // 120
        y0 = sy * (100 + tl) + 220; // 140
        x1 = sx * 100 + 120; // 120
        y1 = sy * 100 + 220; // 140
        // Draw tick
        tft.drawLine(x0, y0, x1, y1, TFT_BLACK);
        // Check if labels should be drawn, with position tweaks
        if (i % 25 == 0) {
            // Calculate label positions
            x0 = sx * (100 + tl + 10) + 120; // 120
            y0 = sy * (100 + tl + 10) + 220; // 140
            switch (i / 25) {
                case -2: tft.drawCentreString("0", x0, y0 - 12, 2); break;
                case -1: tft.drawCentreString("25", x0, y0 - 9, 2); break;
                case 0: tft.drawCentreString("50", x0, y0 - 6, 2); break;
                case 1: tft.drawCentreString("75", x0, y0 - 9, 2); break;
                case 2: tft.drawCentreString("100", x0, y0 - 12, 2); break;
            }
        }
        // Now draw the arc of the scale
        sx = cos((i + 5 - 90) * 0.0174532925);
        sy = sin((i + 5 - 90) * 0.0174532925);
        x0 = sx * 100 + 120; // 120
        y0 = sy * 100 + 220; // 140
        // Draw scale arc, don't draw the last part
        if (i < 50) {
            tft.drawLine(x0, y0, x1, y1, TFT_BLACK);
        }
    }
    tft.drawString("%RH", 195, 180, 2); // Units at bottom right
    tft.drawCentreString("%RH", 120, 140, 4); // Large Center Label
   // tft.drawRect(5, 88, 220, 119, TFT_BLACK); // Draw bottom bezel line
    plotNeedle(0, 0); // Put meter needle at 0
}

// Update needle position
// This function is blocking while needle moves, time depends on ms_delay
// 10ms minimises needle flicker if text is drawn within needle sweep area
// Smaller values OK if text not in sweep area, zero for instant movement but
// does not look realistic... (note: 100 increments for full scale deflection)
// ---------------------------------------------------------------------------
void plotNeedle(int value, byte ms_delay) {
    tft.setTextColor(TFT_BLACK, TFT_WHITE);
    char buf[8]; dtostrf(value, 4, 0, buf);
    tft.drawRightString(buf, 50, 180, 2); // Corrected to 50 & 180 for data humidity
    digital display left value
    if (value < -10) {
        value = -10;    // Limit value to emulate needle end stops
    }
    if (value > 110) {
        value = 110;
    }
    // Move the needle util new value reached
    while (!(value == old_analog)) {
        if (old_analog < value) {
            old_analog++;
        } else {
            old_analog--;
        }
        if (ms_delay == 0) {
            old_analog = value;    // Update immediately id delay is 0
        }
        float sdeg = map(old_analog, -10, 110, -150, -30); // Map value to angle
        // Calcualte tip of needle coords
        float sx = cos(sdeg * 0.0174532925);
        float sy = sin(sdeg * 0.0174532925);
        // Calculate x delta of needle start (does not start at pivot point)
        float tx = tan((sdeg + 90) * 0.0174532925); // 90
        // Erase old needle image
        tft.drawLine(120 + 20 * ltx - 1, 205, osx - 1, osy + 82, TFT_WHITE); // 120
        keep, osy to osy +90
        tft.drawLine(120 + 20 * ltx, 205, osx, osy + 82, TFT_WHITE);
        tft.drawLine(120 + 20 * ltx + 1, 205, osx + 1, osy + 82, TFT_WHITE);
        // Re-plot "RH" text under needle
        tft.setTextColor(TFT_BLACK);
        tft.drawCentreString("%RH", 120, 140, 4); // Changed
        // RePlot Bezel with RH text data and RH label
        // tft.drawRect(20, 174, 220, 30, TFT_BLACK); // Draw bottom bezel line
        // Store new needle end coords for next erase
        ltx = tx;
        osx = sx * 98 + 120;
        osy = sy * 98 + 140;
        // Draw the needle in the new postion, magenta makes needle a bit bolder
        // draws 3 lines to thicken needle
        tft.drawLine(120 + 20 * ltx - 1, 205, osx - 1, osy + 82, TFT_RED); // 120 keep,
        osy to osy +90
        tft.drawLine(120 + 20 * ltx, 205, osx, osy + 82, TFT_MAGENTA);
        tft.drawLine(120 + 20 * ltx + 1, 205, osx + 1, osy + 82, TFT_RED);
        // Slow needle down slightly as it approaches new postion
        if (abs(old_analog - value) < 10) {
            ms_delay += ms_delay / 5;
        }
        // Wait before next update
        delay(ms_delay);
    }
}

// Draw a meter on the screen:
// ---------------------------
void plotLinear(char* label, int x, int y) {
    int w = 36;
    tft.drawRect(x, y, w, 155, TFT_GREY);
    tft.fillRect(x + 2, y + 19, w - 3, 155 - 38, TFT_WHITE);
    tft.setTextColor(TFT_CYAN, TFT_BLACK);
    tft.drawCentreString(label, x + w / 2, y + 2, 2);
    for (int i = 0; i < 110; i += 10) {
        tft.drawFastHLine(x + 20, y + 27 + i, 6, TFT_BLACK);
    }
    for (int i = 0; i < 110; i += 50) {
        tft.drawFastHLine(x + 20, y + 27 + i, 9, TFT_BLACK);
    }
    tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 - 5, TFT_RED);
    tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 + 5, TFT_RED);
    tft.drawCentreString("---", x + w / 2, y + 155 - 18, 2);
}

// Adjust the vertical linear meter pointer positions:
// ---------------------------------------------------
void plotPointer(void) {
    value[0] = int(TempF); // Assign TempF to Value.
    int dy = 187;  // 187
    byte pw = 16;  // 16
    tft.setTextColor(TFT_GREEN, TFT_BLACK);
    // Move the 6 pointers one pixel towards new value
    for (int i = 0; i < 1; i++) { // i < 6
        char buf[8]; dtostrf(value[i], 4, 0, buf); //dtostrf(value[i], 4, 0, buf)
        tft.drawRightString(buf, i * 40 + 287, 207, 2); // Value display (x, y, font
        size)
        int dx = 263 + 40 * i; // Red Pointer "X" position
        if (value[i] < 0) {
            value[i] = 0;    // Limit value to emulate needle end stops
        }
        if (value[i] > 100) {
            value[i] = 100;
        }
        while (!(value[i] == old_value[i])) {
            dy = 180 + 17 - old_value[i]; // Red Pointer "Y" position
            if (old_value[i] > value[i]) {
                tft.drawLine(dx, dy - 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +
                    pw, dy, TFT_WHITE
                    old_value[i]--;
                    tft.drawLine(dx, dy + 6, dx + pw, dy + 1, TFT_RED); //dx, dy + 6, dx +
                    pw, dy + 1, TFT_RED
                } else {
                    tft.drawLine(dx, dy + 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +
                    pw, dy, TFT_WHITE
                    old_value[i]++;
                    tft.drawLine(dx, dy - 6, dx + pw, dy - 1, TFT_RED); //dx, dy + 6, dx +
                    pw, dy + 1, TFT_RED
                }
            }
        }
    }


```