

```

1  /*
2  * This is a program that monitors a DHT sensor for Temperature and Humidity
3  * and feeds a PI Controller that has adjustable setpoints, gains, and humidity
4  * alarm setpoint that also works with many different sensors (just uncomment out
5  * sensor type and change the data pin number if not using the same one "D1")
6  *
7  * The Program Offers the following Functions:
8  * -----
9  * 1) Displays the Temperature and Humidity Locally on Display (analog & digital values)
10 * 2) Contains on-Screen Instructions for setting Up WiFi. (Future)
11 * 3) Displays common error messages for Server and Wifi. (Future)
12 * 4) Detects Sensor failure and displays an error message.
13 * 5) Top Left Button Resets WiFi Settings. (Future)
14 * 6) The Web-Server operates by typing in the Wio Device IP
15 *   Address in a Browser Window and Displays the current
16 *   Temperature and Humidity with an "Auto-Refresh" of Browser
17 *   Every 5 seconds.
18 * 7) Contains a Built in WiFi Manager to connect Wio Device
19 *   to your home router via a Graphical User Interface. (Future)
20 * 8) Provides the following button functionality:
21 *   - Bottom Right switch (Push in) = Menu Operations to set parameters
22 *   - Top Right Button = (+) to adjust menu parameters
23 *   - Top Middle Button = (-) to adjust menu parameters
24 *   - Top Left Button = (Enter) to store menu parameters
25 * 9) Fan Control is PWM controlled through pin "A8"
26 * 10) Parameters are stored in EEPROM and read into program once started.
27 * 11) Contains an internal buzzer when humidity falls below the user setpoint.
28 */
29
30 // Libraries:
31 // -----
32 #include <FlashStorage_SAMD.h>
33 #include <TFT_eSPI.h> // Hardware-specific library
34 #include <SPI.h>
35 #include "DHT.h" // Groove DHT Temperature & Humidity library
36 TFT_eSPI tft = TFT_eSPI(); // Invoke custom library
37
38 // Global Variables:
39 // -----
40 #define TFT_GREY 0x5AEB
41 int count = 0; // Menu Counter
42 int Kp = 50; // Proportional Gain (must be less than 255)
43 int Ki = 5; // Integral Gain (must be less than 255)
44 int address = 100;
45 int flag = 0; // Program Flag to lock menu
46 int Ha = 10; // Humidity Alarm Setpoint
47 int Sp = 70; // Controller Setpoint
48 int PI_Out; // Custom Control Function Return Value
49 const double delta_time = 1.2; // 0.5 Second Sample Rate in Auto (global variable)
50 double I_Term = 0.0; // Integral Term (global variable)
51 double output = 0.0;
52 const double windup_guard = 60.0; // Integral Windup prevention
53 double error = 0.0;
54 double Hum; // Humidity storage Variable
55 double TemperatureC; // Temperature storage variable for Deg C
56 double TempF; // Temperature storage variable for Deg F
57 int Fs; // % Fan Speed
58 // double h = 68; // Test Value, replace with actual humidity reading
59 unsigned long startMillis; // Non Latency Timed Function
60 unsigned long currentMillis;
61 const unsigned long period = 1000; //the value is a number of milliseconds (3 seconds)
62 unsigned long startMillis1; // Non Latency Timed Function
63 unsigned long currentMillis1;
64 const unsigned long period1 = 6000; //the value is a number of milliseconds (6 seconds)
65 #define FLASH_DEBUG 0
66 #define TFT_GREY 0x5AEB
67 #define LOOP_PERIOD 35 // Display updates every 35 ms
68 float ltx = 0; // Saved x coord of bottom of needle
69 uint16_t osx = 120, osy = 120; // Saved x & y coords (osx = 120, osy = 120)

```

```

70 uint32_t updateTime = 0; // time for next update
71 int old_analog = -999; // Value last displayed
72 int old_digital = -999; // Value last displayed
73 int value[6] = {0, 0, 0, 0, 0, 0};
74 int old_value[6] = { -1, -1, -1, -1, -1, -1};
75 int d = 0;
76 boolean interlock = true; // Stops Program execution while in Menu
77
78 // DHT Sensor Characteristics (Uncomment whatever type you're using)
79 // -----
80 // #define DHTTYPE DHT11 // DHT 11
81 #define DHTTYPE DHT22 // DHT 22 (AM2302)
82 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
83 // #define DHTTYPE DHT10 // DHT 10
84 // #define DHTTYPE DHT20 // DHT 20
85 #define DHTPIN D1 // Data Pin we're connected to
86 DHT dht(DHTPIN, DHTTYPE); // DHT11 DHT21 DHT22
87 // DHT dht(DHTTYPE); // DHT10 DHT20 don't need to define Pin
88
89 // Motor Drive Pin:
90 // -----
91 #define PWM_Pin A8 // Motor Drive Pin
92
93 // Main Program:
94 // =====
95
96 void setup() {
97   Serial.begin(115200);
98   tft.init();
99   tft.setRotation(3);
100  // tft.setTextSize(2);
101  tft.fillScreen(TFT_BLACK);
102  tft.setTextColor(TFT_WHITE);
103  tft.drawString("Cigar Humidor Controller", 10, 10, 4); // prints strings from (x, y,
   font size)
104  tft.drawString("With Advanced Features", 10, 50, 4);
105  tft.drawString("By: Roy H Guerra Jr.", 10, 90, 4);
106  pinMode(WIO_5S_UP, INPUT_PULLUP); // Enable Wio Button puulup Resistors
107  pinMode(WIO_5S_DOWN, INPUT_PULLUP);
108  pinMode(WIO_5S_LEFT, INPUT_PULLUP);
109  pinMode(WIO_5S_RIGHT, INPUT_PULLUP);
110  pinMode(WIO_5S_PRESS, INPUT_PULLUP);
111  pinMode(WIO_KEY_A, INPUT_PULLUP);
112  pinMode(WIO_KEY_B, INPUT_PULLUP);
113  pinMode(WIO_KEY_C, INPUT_PULLUP);
114  pinMode(PWM_Pin, OUTPUT); // PWM Channel
115  pinMode(WIO_BUZZER, OUTPUT); // Internal Wio Buzzer
116  dht.begin(); // Initialize DHT sensor
117  delay(2000); // 2S loop delay
118  tft.fillScreen(TFT_BLACK);
119  updateTime = millis(); // Next update time
120  startMillis = millis(); // initial time stamp
121  startMillis1 = millis(); // initial time stamp
122  analogMeter(); // Draw analog meter
123  plotLinear("oF", 260, 70); // Draw 1 linear meters
124 }
125
126 void loop() {
127   currentMillis = millis(); // Get a time Stamp
128   currentMillis1 = millis(); // Get a time Stamp
129   if (digitalRead(WIO_5S_PRESS) == LOW) {
130     Serial.println("5 Way Button Press");
131     interlock = false; // Set interlock
132     count = 1; // Set Counter
133     Serial.println("Count = " + String(count));
134   }
135   switch (count) {
136     case 1:
137       tft.fillScreen(TFT_BLACK);

```

```

138     tft.setTextColor(TFT_CYAN);
139     tft.drawString("Set Humidity Alarm SP", 10, 10, 4); //prints strings from (x, y,
font size)
140     tft.drawString("-----", 10, 30, 4);
141     tft.setTextColor(TFT_YELLOW);
142     tft.drawString("Press Top Right Button (+)", 10, 70, 4);
143     tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
144     tft.setTextColor(TFT_WHITE);
145     tft.drawString("Humidity Alarm SP = ", 10, 160, 4);
146     tft.drawRect(245,150,55,35,TFT_WHITE);
147     tft.drawString(String(Ha), 250, 160, 4);
148     tft.setTextColor(TFT_RED);
149     tft.drawString("Press Top Left Button To", 10, 192, 4);
150     tft.drawString("Save Configuration (exit)", 10, 215, 4);
151     flag = 1; // Change program flag
152     while (flag == 1) {
153         if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 1)){
154             Serial.println("B Key pressed");
155             if (Ha > 0){
156                 Ha -= 1;
157                 tft.fillRect(245,150,55,35,TFT_BLACK);
158                 tft.drawRect(245,150,55,35,TFT_WHITE);
159                 tft.setTextColor(TFT_WHITE);
160                 tft.drawString(String(Ha), 250, 160, 4);
161             }
162             Serial.println("Ha = " + String(Ha));
163         }
164         if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 1)) {
165             Serial.println("A Key pressed");
166             if (Ha < 80){
167                 Ha += 1;
168                 tft.fillRect(245,150,55,35,TFT_BLACK);
169                 tft.drawRect(245,150,55,35,TFT_WHITE);
170                 tft.setTextColor(TFT_WHITE);
171                 tft.drawString(String(Ha), 250, 160, 4);
172             }
173             Serial.println("Ha = " + String(Ha));
174         }
175         if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 1)) {
176             Serial.println("C Key pressed");
177             /**
178             The function EEPROM.update(address, val) is equivalent to the following:
179             if( EEPROM.read(address) != val ) {
180                 EEPROM.write(address, val);
181             }
182             ***/
183             EEPROM.update(address + 20, Ha);
184             EEPROM.commit();
185             Serial.println("Wrote " + String(Ha) + " To EEPROM Address 120");
186             tft.fillScreen(TFT_BLACK);
187             tft.setTextColor(TFT_CYAN);
188             tft.drawString("Humidity Alarm SP Saved", 10, 70, 4);
189             delay(2000); // 2S loop delay
190             tft.fillScreen(TFT_BLACK);
191             flag = 0; // Reset flag
192             count = 2; // Next menu Option
193         }
194         delay(150); // 150mS loop delay
195     }
196     break;
197 case 2:
198     tft.fillScreen(TFT_BLACK);
199     tft.setTextColor(TFT_MAGENTA);
200     tft.drawString("Set Controller SP", 10, 10, 4); //prints strings from (x, y, font
size)
201     tft.drawString("-----", 10, 30, 4);
202     tft.setTextColor(TFT_YELLOW);
203     tft.drawString("Press Top Right Button (+)", 10, 70, 4);
204     tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);

```

```

205     tft.setTextColor(TFT_WHITE);
206     tft.drawString("    Controller SP = ", 10, 160, 4);
207     tft.drawRect(245,150,55,35,TFT_WHITE);
208     tft.drawString(String(Sp), 250, 160, 4);
209     tft.setTextColor(TFT_RED);
210     tft.drawString("Press Top Left Button To", 10, 192, 4);
211     tft.drawString("Save Configuration (exit)", 10, 215, 4);
212     flag = 1; // Change program flag
213     while (flag == 1) {
214         if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 2)){
215             Serial.println("B Key pressed");
216             if (Sp > 50){
217                 Sp -= 1;
218                 tft.fillRect(245,150,55,35,TFT_BLACK);
219                 tft.drawRect(245,150,55,35,TFT_WHITE);
220                 tft.setTextColor(TFT_WHITE);
221                 tft.drawString(String(Sp), 250, 160, 4);
222             }
223             Serial.println("Ha = " + String(Sp));
224         }
225         if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 2)) {
226             Serial.println("A Key pressed");
227             if (Sp < 80){
228                 Sp += 1;
229                 tft.fillRect(245,150,55,35,TFT_BLACK);
230                 tft.drawRect(245,150,55,35,TFT_WHITE);
231                 tft.setTextColor(TFT_WHITE);
232                 tft.drawString(String(Sp), 250, 160, 4);
233             }
234             Serial.println("Ha = " + String(Sp));
235         }
236         if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 2)) {
237             Serial.println("C Key pressed");
238             /**
239             The function EEPROM.update(address, val) is equivalent to the following:
240             if( EEPROM.read(address) != val ) {
241                 EEPROM.write(address, val);
242             }
243             ***/
244             EEPROM.update(address + 30, Sp);
245             EEPROM.commit();
246             Serial.println("Wrote " + String(Sp) + " To EEPROM Address 130");
247             tft.fillScreen(TFT_BLACK);
248             tft.setTextColor(TFT_MAGENTA);
249             tft.drawString("Controller Setpoint Saved", 10, 70, 4);
250             delay(2000); // 2S loop delay
251             tft.fillScreen(TFT_BLACK);
252             flag = 0; // Reset flag
253             count = 3; // Next menu Option
254         }
255         delay(150); // 150mS loop delay
256     }
257     break;
258 case 3:
259     tft.fillScreen(TFT_BLACK);
260     tft.setTextColor(TFT_GREEN);
261     tft.drawString("Set Proportional Gain", 10, 10, 4); //prints strings from (x, y,
font size)
262     tft.drawString("-----", 10, 30, 4);
263     tft.setTextColor(TFT_YELLOW);
264     tft.drawString("Press Top Right Button (+)", 10, 70, 4);
265     tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
266     tft.setTextColor(TFT_WHITE);
267     tft.drawString("Proportional Gain = ", 10, 160, 4);
268     tft.drawRect(245,150,55,35,TFT_WHITE);
269     tft.drawString(String(Kp), 250, 160, 4);
270     tft.setTextColor(TFT_RED);
271     tft.drawString("Press Top Left Button To", 10, 192, 4);
272     tft.drawString("Save Configuration (exit)", 10, 215, 4);

```

```

273 flag = 1; // Change program flag
274 while (flag == 1) {
275     if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 3)){
276         Serial.println("B Key pressed");
277         if (Kp > 1){
278             Kp -= 1;
279             tft.fillRect(245,150,55,35,TFT_BLACK);
280             tft.drawRect(245,150,55,35,TFT_WHITE);
281             tft.setTextColor(TFT_WHITE);
282             tft.drawString(String(Kp), 250, 160, 4);
283         }
284         Serial.println("Kp = " + String(Kp));
285     }
286     if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 3)) {
287         Serial.println("A Key pressed");
288         if (Kp < 100){
289             Kp += 1;
290             tft.fillRect(245,150,55,35,TFT_BLACK);
291             tft.drawRect(245,150,55,35,TFT_WHITE);
292             tft.setTextColor(TFT_WHITE);
293             tft.drawString(String(Kp), 250, 160, 4);
294         }
295         Serial.println("Kp = " + String(Kp));
296     }
297     if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 3)) {
298         Serial.println("C Key pressed");
299         /**
300         The function EEPROM.update(address, val) is equivalent to the following:
301         if( EEPROM.read(address) != val ) {
302             EEPROM.write(address, val);
303         }
304         ***/
305         EEPROM.update(address, Kp);
306         EEPROM.commit();
307         Serial.println("Wrote " + String(Kp) + " To EEPROM Address 100");
308         tft.fillScreen(TFT_BLACK);
309         tft.setTextColor(TFT_GREEN);
310         tft.drawString("Proportional Gain Saved", 10, 70, 4);
311         delay(2000); // 2S loop delay
312         tft.fillScreen(TFT_BLACK);
313         flag = 0; // Reset flag
314         count = 4; // Next menu Option
315     }
316     delay(150); // 150mS loop delay
317 }
318 break;
319 case 4:
320     tft.fillScreen(TFT_BLACK);
321     tft.setTextColor(TFT_BLUE);
322     tft.drawString("Set Integral Gain", 10, 10, 4); //prints strings from (x, y, font
size)
323     tft.drawString("-----", 10, 30, 4);
324     tft.setTextColor(TFT_YELLOW);
325     tft.drawString("Press Top Right Button (+)", 10, 70, 4);
326     tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
327     tft.setTextColor(TFT_WHITE);
328     tft.drawString("    Integral Gain = ", 10, 160, 4);
329     tft.drawRect(245,150,55,35,TFT_WHITE);
330     tft.drawString(String(Ki), 250, 160, 4);
331     tft.setTextColor(TFT_RED);
332     tft.drawString("Press Top Left Button To", 10, 192, 4);
333     tft.drawString("Save Configuration (exit)", 10, 215, 4);
334     flag = 1; // Change program flag
335     while (flag == 1) {
336         if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 4)){
337             Serial.println("B Key pressed");
338             if (Ki > 0){
339                 Ki -= 1;
340                 tft.fillRect(245,150,55,35,TFT_BLACK);

```

```

341         tft.drawRect(245,150,55,35,TFT_WHITE);
342         tft.setTextColor(TFT_WHITE);
343         tft.drawString(String(Ki), 250, 160, 4);
344     }
345     Serial.println("Ki = " + String(Ki));
346 }
347 if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 4)) {
348     Serial.println("A Key pressed");
349     if (Ki < 50){
350         Ki += 1;
351         tft.fillRect(245,150,55,35,TFT_BLACK);
352         tft.drawRect(245,150,55,35,TFT_WHITE);
353         tft.setTextColor(TFT_WHITE);
354         tft.drawString(String(Ki), 250, 160, 4);
355     }
356     Serial.println("Ki = " + String(Ki));
357 }
358 if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 4)) {
359     Serial.println("C Key pressed");
360     /**
361     The function EEPROM.update(address, val) is equivalent to the following:
362     if( EEPROM.read(address) != val ) {
363         EEPROM.write(address, val);
364     }
365     ***/
366     EEPROM.update(address + 10, Ki);
367     EEPROM.commit();
368     Serial.println("Wrote " + String(Ki) + " To EEPROM Address 110");
369     tft.fillRect(TFT_BLACK);
370     tft.setTextColor(TFT_BLUE);
371     tft.drawString("Integral Gain Saved", 10, 70, 4);
372     delay(2000); // 2S loop delay
373     tft.fillRect(TFT_BLACK);
374     flag = 0; // Reset flag
375     interlock = true; // Reset Interlock
376     count = 0; // Next menu Option
377     NVIC_SystemReset(); // Re-Start Program
378 }
379     delay(150); // 150mS loop delay
380 }
381 break;
382 default:
383     count = 0; // Default
384 break;
385 }
386 // delay(100); // 100mS loop delay
387 if (interlock == true){
388     // No Operation of Program past this point once interlock is set while in menu's
389     if (currentMillis - startMillis >= period){ // Test whether the period has elapsed
390         SensorData(); // Goto Function
391         controller(); // Goto Function
392         // Plot Fan Drive
393         tft.setTextColor(TFT_MAGENTA); // Text color
394         tft.drawString("Fan Drive = ", 55, 58, 2); // SCREEN Header
395         tft.fillRect(135,56,48,20,TFT_BLACK); // Draw a Rect to erase previous data
396         tft.drawRect(135,56,48,20,TFT_MAGENTA); // Draw a Rect.
397         tft.drawString(String(Fs) + " %", 140, 58, 2); //prints strings from (x, y)
398         startMillis = currentMillis; // New Time Stamp
399     }
400     if (currentMillis1 - startMillis1 >= period1){ // Test whether the period has elapsed
401         buzzer(); // Goto Function
402         startMillis1 = currentMillis1; // New Time Stamp
403     }
404     if (updateTime <= millis()) {
405         updateTime = millis() + LOOP_PERIOD;
406         //value[0] = map(analogRead(A0), 0, 1023, 0, 100); // Test with an actual value
407         // value[0] = 50 + 50 * sin((d + 0) * 0.0174532925); // Create a Sine wave for
testing
408         plotPointer(); // Goto Function

```

```

409     plotNeedle(int(Hum), 0); // Goto Function
410 }
411 }
412 }
413
414 // PI Controller Function:
415 // -----
416 float Calculate_PI () {
417     // Read EEPROM Kp & Ki, Ha, & Sp:
418     Kp = EEPROM.read(address);
419     Serial.println("Kp = " + String(Kp));
420     Ki = EEPROM.read(address + 10);
421     Serial.println("Ki = " + String(Ki));
422     Ha = EEPROM.read(address + 20);
423     Serial.println("Ha = " + String(Ha));
424     Sp = EEPROM.read(address + 30);
425     Serial.println("Sp = " + String(Sp));
426     if ((Kp == 255) || (Ki == 255) || (Sp == 255) || (Ha == 255)) { // Guards against
EEPROM not being set
427         Kp = 50;
428         Ki = 5;
429         Ha = 50;
430         Sp = 70;
431     }
432     error = Sp - Hum; // Error Term, h = feedback
433     I_Term += (error * delta_time); // Intergral Term
434     if (I_Term > windup_guard){ // Positive Integral Windup Guard
435         I_Term = windup_guard;
436     }
437     if (I_Term < - windup_guard){ // Negative Integral Windup Guard
438         I_Term = - windup_guard;
439     }
440     if (isnan(I_Term)){ // Reset if NAN
441         I_Term = 0;
442     }
443     output = (Kp * error) + (Ki * I_Term); // Controller Output (Proportional + Integral)
444     output = constrain(output, 0, 255); // Limits Controller Range
445     Serial.println("Kp = " + String(Kp)); // Debug
446     Serial.println("Ki = " + String(Ki)); // Debug
447     Serial.println("Setpoint = " + String(Sp)); // Debug
448     Serial.println("Feedback (humidity) = " + String(Hum)); // Debug
449     Serial.println("Error = " + String(error)); // Debug
450     Serial.println("I_Term = " + String(I_Term)); // Debug
451     Serial.println("Ki *I_Term = " + String(Ki * I_Term)); // Debug
452     Serial.println("P_Term = " + String(Kp * error)); // Debug
453     Serial.println("Output = " + String(output)); // Debug
454     Serial.println("Alarm Setpoint = " + String(Ha)); // Debug
455     return int(output); // Return PI Control Value as an integer
456 }
457
458 // Function to Sound Buzzer:
459 // -----
460 void buzzer(){ // Buzzer Function Block
461     if (Hum < Ha){
462         analogWrite(WIO_BUZZER, 128);
463         delay(1000);
464         analogWrite(WIO_BUZZER, 0);
465         delay(1000);
466     }
467 }
468
469 // Function to Read Control Loop and set PWM and Speed Indication:
470 // -----
471 void controller(){ // Controller Function Block
472     PI_Out = Calculate_PI(); // Calculate new PI Control Value
473     Serial.println("PI_Out = " + String(PI_Out)); // Debug
474     analogWrite(PWM_Pin, PI_Out); // PWM Value (0-255)
475     Fs = map(output, 0, 255, 0, 100); // Rescale controller output to % fan speed
476     Serial.println("Fan Speed = " + String(Fs)); // Debug

```

```

477 }
478
479 // Read Sensor Function
480 // -----
481 void SensorData(){
482     Hum = dht.readHumidity(); // Measure the humidity
483     Serial.println("Humidity = " + String(Hum));
484     TemperatureC = dht.readTemperature(); // Measure the temperature
485     TempF = ((TemperatureC * 9/5) + 32); // Convert temperature to degrees Fahrenheit
486     Serial.println("Temperature = " + String(TempF));
487     // Compare temperature & humidity events and perform a check sum.
488     if (isnan(TemperatureC) || isnan(Hum)){ // Print "0" for a bad reading
489         TempF = 0;
490         Hum = 0;
491         Serial.println("Bad Connection or Sensor");
492     }
493 }
494
495 // Draw the Horizontal Analog Meter & Menu on the screen
496 // -----
497 void analogMeter() {
498     // Meter outline
499     tft.fillRect(0, 85, 239, 126, TFT_GREY); // 0, 0, 239, 126 (x, y, w, h)
500     tft.fillRect(5, 88, 230, 119, TFT_WHITE); // 5, 3, 230, 119,
501     //tft.fillRect(5, 10, 100, 50, TFT_WHITE); // SCREEN Header
502     tft.setTextColor(TFT_WHITE);
503     tft.drawString(" Cigar Humidor Parameters", 5, 10, 4); // SCREEN Header
504     tft.drawString(" -----", 5, 35, 4); // SCREEN
Header
505     tft.setTextColor(TFT_BLACK); // Text color
506     // Draw ticks every 5 degrees from -50 to +50 degrees (100 deg. FSD swing)
507     for (int i = -50; i < 51; i += 5) {
508         // Long scale tick length
509         int t1 = 15;
510         // Coodinates of tick to draw
511         float sx = cos((i - 90) * 0.0174532925);
512         float sy = sin((i - 90) * 0.0174532925);
513         uint16_t x0 = sx * (100 + t1) + 120; // 120
514         uint16_t y0 = sy * (100 + t1) + 220; // 140
515         uint16_t x1 = sx * 100 + 120; // 120
516         uint16_t y1 = sy * 100 + 220; // 140
517         // Coordinates of next tick for zone fill
518         float sx2 = cos((i + 5 - 90) * 0.0174532925);
519         float sy2 = sin((i + 5 - 90) * 0.0174532925);
520         int x2 = sx2 * (100 + t1) + 120; // 120
521         int y2 = sy2 * (100 + t1) + 220; // 140
522         int x3 = sx2 * 100 + 120; // 120
523         int y3 = sy2 * 100 + 220; // 140
524         // Yellow zone limits
525         if (i >= -50 && i < 1) {
526             tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_YELLOW);
527             tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_YELLOW);
528         }
529         // Green zone limits
530         if (i >= 1 && i < 25) { // 0
531             tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_GREEN);
532             tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_GREEN);
533         }
534         // Orange zone limits
535         if (i >= 25 && i < 50) {
536             tft.fillTriangle(x0, y0, x1, y1, x2, y2, TFT_ORANGE);
537             tft.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_ORANGE);
538         }
539         // Short scale tick length
540         if (i % 25 != 0) {
541             t1 = 8;
542         }
543         // Recalculate coords incase tick lenght changed
544         x0 = sx * (100 + t1) + 120; // 120

```

```

545     y0 = sy * (100 + t1) + 220; // 140
546     x1 = sx * 100 + 120; // 120
547     y1 = sy * 100 + 220; // 140
548     // Draw tick
549     tft.drawLine(x0, y0, x1, y1, TFT_BLACK);
550     // Check if labels should be drawn, with position tweaks
551     if (i % 25 == 0) {
552         // Calculate label positions
553         x0 = sx * (100 + t1 + 10) + 120; // 120
554         y0 = sy * (100 + t1 + 10) + 220; // 140
555         switch (i / 25) {
556             case -2: tft.drawCentreString("", x0, y0 - 12, 2); break;
557             case -1: tft.drawCentreString("25", x0, y0 - 9, 2); break;
558             case 0: tft.drawCentreString("50", x0, y0 - 6, 2); break;
559             case 1: tft.drawCentreString("75", x0, y0 - 9, 2); break;
560             case 2: tft.drawCentreString("100", x0, y0 - 12, 2); break;
561         }
562     }
563     // Now draw the arc of the scale
564     sx = cos((i + 5 - 90) * 0.0174532925);
565     sy = sin((i + 5 - 90) * 0.0174532925);
566     x0 = sx * 100 + 120; // 120
567     y0 = sy * 100 + 220; // 140
568     // Draw scale arc, don't draw the last part
569     if (i < 50) {
570         tft.drawLine(x0, y0, x1, y1, TFT_BLACK);
571     }
572 }
573 tft.drawString("%RH", 195, 180, 2); // Units at bottom right
574 tft.drawCentreString("%RH", 120, 140, 4); // Large Center Label
575 // tft.drawRect(5, 88, 220, 119, TFT_BLACK); // Draw bottom bezel line
576 plotNeedle(0, 0); // Put meter needle at 0
577 }
578
579 // Update needle position
580 // This function is blocking while needle moves, time depends on ms_delay
581 // 10ms minimises needle flicker if text is drawn within needle sweep area
582 // Smaller values OK if text not in sweep area, zero for instant movement but
583 // does not look realistic... (note: 100 increments for full scale deflection)
584 // -----
585 void plotNeedle(int value, byte ms_delay) {
586     tft.setTextColor(TFT_BLACK, TFT_WHITE);
587     char buf[8]; dtostrf(value, 4, 0, buf);
588     tft.drawRightString(buf, 50, 180, 2); // Corrected to 50 & 180 for data humidity
589     // digital display left value
590     if (value < -10) {
591         value = -10; // Limit value to emulate needle end stops
592     }
593     if (value > 110) {
594         value = 110;
595     }
596     // Move the needle util new value reached
597     while (!(value == old_analog)) {
598         if (old_analog < value) {
599             old_analog++;
600         } else {
601             old_analog--;
602         }
603         if (ms_delay == 0) {
604             old_analog = value; // Update immediately id delay is 0
605         }
606         float sdeg = map(old_analog, -10, 110, -150, -30); // Map value to angle
607         // Calcualte tip of needle coords
608         float sx = cos(sdeg * 0.0174532925);
609         float sy = sin(sdeg * 0.0174532925);
610         // Calculate x delta of needle start (does not start at pivot point)
611         float tx = tan((sdeg + 90) * 0.0174532925); // 90
612         // Erase old needle image
613         tft.drawLine(120 + 20 * ltx - 1, 205, osx - 1, osy + 82, TFT_WHITE); // 120

```

```

        keep, osy to osy +90
613 tft.drawLine(120 + 20 * ltx, 205, osx, osy + 82, TFT_WHITE);
614 tft.drawLine(120 + 20 * ltx + 1, 205, osx + 1, osy + 82, TFT_WHITE);
615 // Re-plot "RH" text under needle
616 tft.setTextColor(TFT_BLACK);
617 tft.drawCentreString("%RH", 120, 140, 4); // Changed
618 // RePlot Bezel with RH text data and RH label
619 // tft.drawRect(20, 174, 220, 30, TFT_BLACK); // Draw bottom bezel line
620 // Store new needle end coords for next erase
621 ltx = tx;
622 osx = sx * 98 + 120;
623 osy = sy * 98 + 140;
624 // Draw the needle in the new postion, magenta makes needle a bit bolder
625 // draws 3 lines to thicken needle
626 tft.drawLine(120 + 20 * ltx - 1, 205, osx - 1, osy + 82, TFT_RED); // 120 keep,
    osy to osy +90
627 tft.drawLine(120 + 20 * ltx, 205, osx, osy + 82, TFT_MAGENTA);
628 tft.drawLine(120 + 20 * ltx + 1, 205, osx + 1, osy + 82, TFT_RED);
629 // Slow needle down slightly as it approaches new postion
630 if (abs(old_analog - value) < 10) {
631     ms_delay += ms_delay / 5;
632 }
633 // Wait before next update
634 delay(ms_delay);
635 }
636 }
637
638 // Draw a meter on the screen:
639 // -----
640 void plotLinear(char* label, int x, int y) {
641     int w = 36;
642     tft.drawRect(x, y, w, 155, TFT_GREY);
643     tft.fillRect(x + 2, y + 19, w - 3, 155 - 38, TFT_WHITE);
644     tft.setTextColor(TFT_CYAN, TFT_BLACK);
645     tft.drawCentreString(label, x + w / 2, y + 2, 2);
646     for (int i = 0; i < 110; i += 10) {
647         tft.drawFastHLine(x + 20, y + 27 + i, 6, TFT_BLACK);
648     }
649     for (int i = 0; i < 110; i += 50) {
650         tft.drawFastHLine(x + 20, y + 27 + i, 9, TFT_BLACK);
651     }
652     tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 - 5, TFT_RED);
653     tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 + 5, TFT_RED);
654     tft.drawCentreString("---", x + w / 2, y + 155 - 18, 2);
655 }
656
657 // Adjust the vertical linear meter pointer positions:
658 // -----
659 void plotPointer(void) {
660     value[0] = int(TempF); // Assign TempF to Value.
661     int dy = 187; // 187
662     byte pw = 16; // 16
663     tft.setTextColor(TFT_GREEN, TFT_BLACK);
664     // Move the 6 pointers one pixel towards new value
665     for (int i = 0; i < 6; i++) { // i < 6
666         char buf[8]; dtostrf(value[i], 4, 0, buf); //dtostrf(value[i], 4, 0, buf)
667         tft.drawRightString(buf, i * 40 + 287, 207, 2); // Value display (x, y, font
            size)
668         int dx = 263 + 40 * i; // Red Pointer "X" position
669         if (value[i] < 0) {
670             value[i] = 0; // Limit value to emulate needle end stops
671         }
672         if (value[i] > 100) {
673             value[i] = 100;
674         }
675         while (!(value[i] == old_value[i])) {
676             dy = 180 + 17 - old_value[i]; // Red Pointer "Y" position
677             if (old_value[i] > value[i]) {
678                 tft.drawLine(dx, dy - 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +

```

```
        pw, dy, TFT_WHITE
679     old_value[i]--;
680     tft.drawLine(dx, dy + 6, dx + pw, dy + 1, TFT_RED); //dx, dy + 6, dx +
        pw, dy + 1, TFT_RED
681     } else {
682     tft.drawLine(dx, dy + 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +
        pw, dy, TFT_WHITE
683     old_value[i]++;
684     tft.drawLine(dx, dy - 6, dx + pw, dy - 1, TFT_RED); //dx, dy + 6, dx +
        pw, dy + 1, TFT_RED
685     }
686     }
687     }
688     }
689
690
```