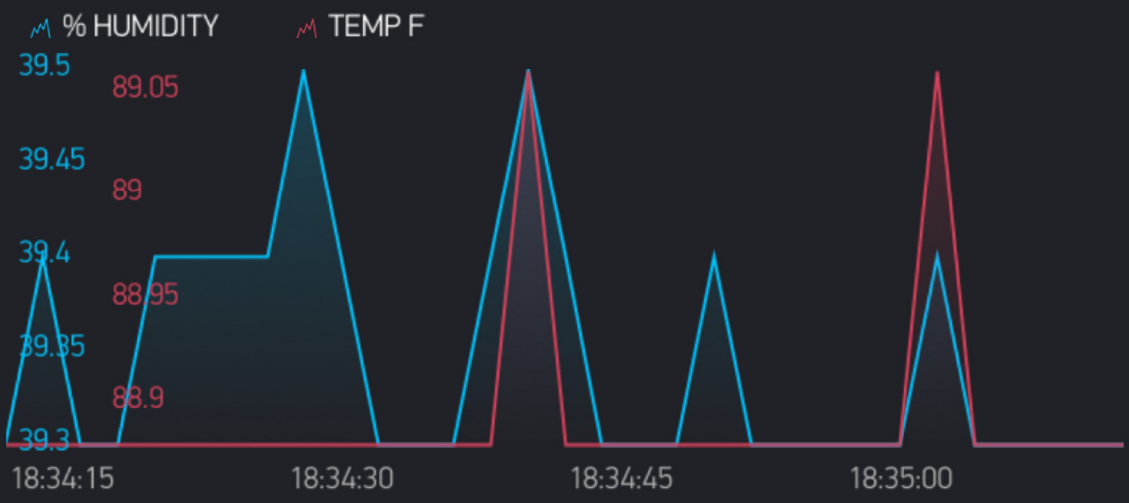
Cigar Humidor

HUMIDITY. (%RH)

TEMPERATURE (DEG.F)



Live 1h 6h 1d 1w 1M 3M

HUMIDITY CONTROL SETPOINT 70.8

HUMIDITY ALARM SETPOINT (%RH < ALARM SETPOINT) 20.2

PROPORTIONAL GAIN 707.3

INTEGRAL GAIN 50.8

ESP8266 MQTT Cigar Humidor Tempaerature and Humidity Publisher
Uses distilled water in a container with a Muffin fan that keeps the Humidity level at the setpoint using a built in Proportional and Integral Control Algorithm. The fan is PWM controlled by the output of the Controller.

5VDC Muffin Fan

Note - Place this in Cigar Humidor

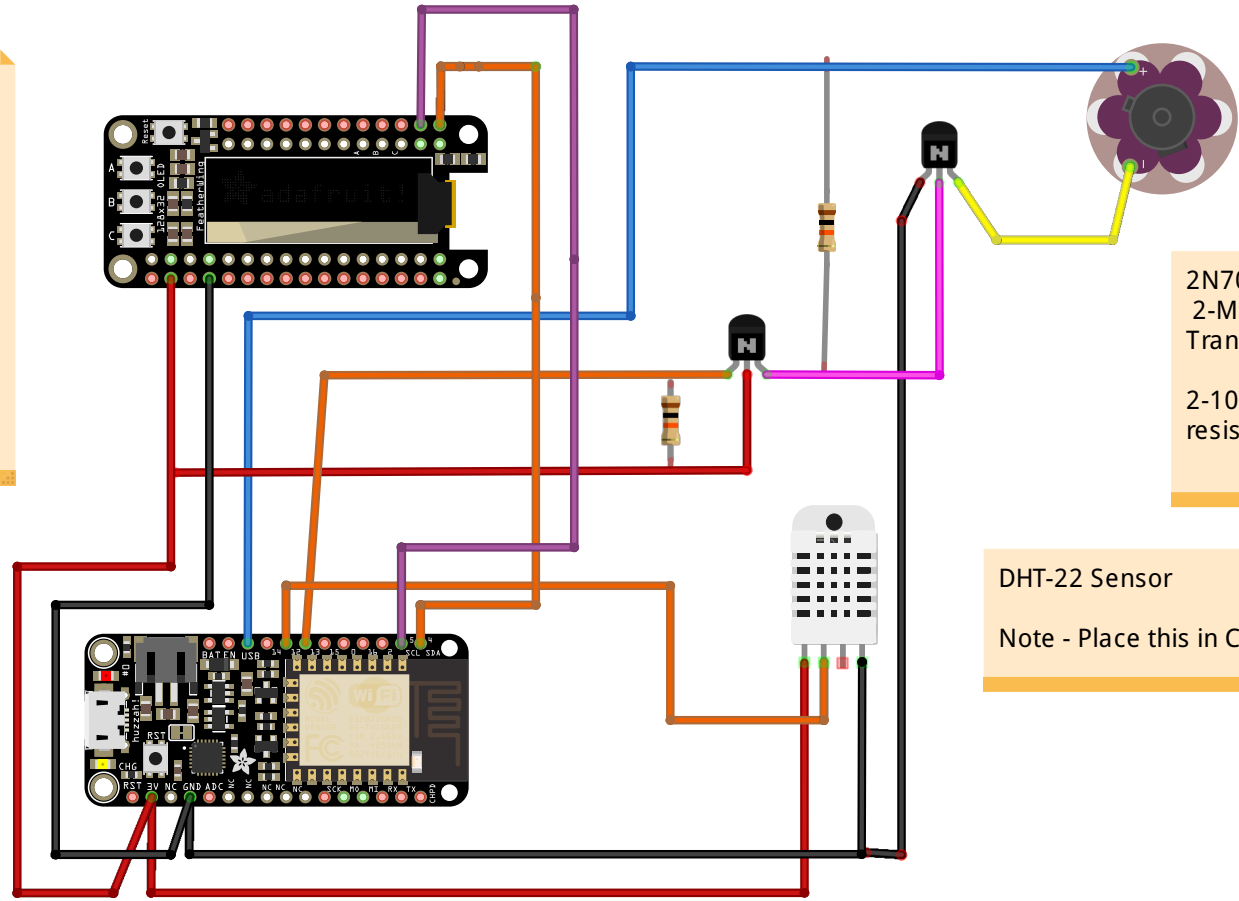
Switch 'B' Resets Wifi (By deleting 'wifi.dat' File, and placing control systtem in a fixed Auto mode with a fixed humidity setpoint of 70% with no cloud server contols)

Note - Do this if Network Server goes down.

On-Board 'blue" LED lights up when wifi connection is established.

OLED Display is (128X32)

WiFi Board is ESP8266 (Adafruit Feather)



2N7000
2-MOSFET Transistors

2-10K resistors

DHT-22 Sensor

Note - Place this in Cigar Humidor

Initial WiFi Setup:

1) Plug in, and turn on power.
2) Press the “reset” button on the device.
3) If WiFi was previously setup, skip this section, otherwise continue.
4) If there was no previous WiFi setup, the local OLED Display will display the following message “Starting Access Point”; “Establish network to > Jon Humidor”; “Use phone or computer”. Note – this screen does not appear if there was a previously established WiFi connection to the same network.
5) Using your phone go to WiFi settings or if using a laptop computer for to network settings and choose “Jon Humidor”.
6) When the WiFi configuration portal screen comes up select “Configure WiFi” Option only.
7) Select your home network (should be underlined), then enter your passcode in the required field and then select “save”
8) Once the device connects to your home network, the on board “blue” LED illuminates, and your home network “SSID” and “IP Address” given to the device appears on the OLED Display, then goes away after 2 seconds.
9) At this point the device is set up and will display the humidity and the temperature of the cigar humidor on the phone AP and locally on the OLED Display (switches between humidity and temperature). Note - The OLED blanks out after every cycle so the display does not get burned in, but restarts.
10) Your current WiFi settings are stored, and upon a loss of power the device will automatically connect when the power returns using the same settings before power was lost.

Humidity PI Controller Setup:

1) Go to the “Blynk” phone AP, and start with the following initial conditions by adjusting the sliders:
   - Humidity Control Setpoint = 70
   - Humidity Alarm Setpoint = 20
   - Proportional Gain = 700
   - Integral Gain = 50
2) Allow 4-8 hours for the system to stabilize, and readjust the settings in the previous step to fine tune (as required).
3) The humidity Alarm Setpoint is on a 10 minute timer, so it can take up to 10 minutes to get a notification and E-Mail if the (humidity < humidity alarm setpoint).
4) For faster response and tighter controller operation, increase the proportional gain.
5) If the setpoint is not reached or slightly overshoots, increase the integral gain.
6) For best tuning results, look at the “Chart” on the phone AP and choose “1Hour” or “6Hours” of data and look at the humidity variance band (high – low). Adjust the gains to get the smallest variance humidity band. Note best control band response is limited by the accuracy of the sensor which is +/- 2% to 5% typical.

Non Phone AP Manual Operation:

1) To use the controller without the phone AP, and to keep the controller operating in “Auto Operation” with a fixed setpoint of 70% relative humidity perform the following steps.
2) Depress and release the “reset” button, then quickly go to “Button-B” and hold down until a message is displayed “manual mode of operation”; “press reset to return”. This step will also delete the stored WiFi settings.
3) To return to the Phone AP, depress and release the “reset” button, and proceed to the “initial WiFi setup” section of this document.

Sensor Hardware / Wiring Detection

1) If the local temperature or humidity reading on the OLED Display says “nan” which stands for “not a number”, the sensor could be defective or you may have a wiring or hardware issue.
2) This condition is further exemplified by a separate message on the OLED Display which says “bad check sum”; “check sensor”.
3) The phone AP display for humidity and temperature shows “-----” on each gauge, and the fan switches to full on to keep humidity.

To Erase the Stored WiFi Settings or set up a new WiFi Connection

1) If your network information changes, using your phone or laptop computer go to WiFi settings or network settings on a laptop and connect to “Jon’s Humidor”, and start with step #5 of the “initial WiFi setup”
2) To erase the stored WiFi settings go to the “Non Phone AP Manual Operation” section of this document and follow directions.

```
/*
ESP8266 Program for a Cigar Humidor Control with BLYNK:

Hardware Required:
-----------------
- Adafruit ESP8266 Feather
- Adafruit OLED Feather Wing Display Module using IIC protocol

The circuit has the following features:
 0) Starts as an Access Point, and connects to network via a phone or laptop through a GUI.
 1) Connects and Transmits / Recieves through the house wireless router to a Network server.
 2) Connects to the Blynk cloud server webpage that could be pulled up anywhere in the world.
 3) The "blue LED" lights when connected to Wi-Fi.
 4) Auto syncs back to last values on server when power is lost.
 5) Sends notifications and E-mails when Humidity is below alarm setpoint slider
 6) Displays humidity and temperature on large dial gauges
 7) Has a graphing chart that trends humidity and temperature values
 8) Has sliders for humidity setpoint, alarm setpoint, proportional gain and integral gain
 9) Has manual overide operation in case network server is lost

Note(s):
 * First download the Blynk App on your phone, and set up your account and Project.
 * Don't put Blynk.virtualWrite and any other Blynk.* command inside void loop() the connection will be terminated.
 * Call functions with intervals. For example, this SimpleTimer Library is a library for timed events.
 * Avoid using long delays with delay() - it may cause connection breaks;
 * If you send more than 100 values per second - you may cause Flood Error and your hardware will be disconnected from the server.
 * Be careful sending a lot of Blynk.virtualWrite commands as most hardware is not very powerful (like ESP8266) so it may not handle many requests.
 * When first connecting to a network, go to Wi-Fi settings when already connected to the house router, and choose "Window Shade" to set up.

OLED Wiring
-----------
 * VDD  = + 3v of Feather Wing
 * Gnd = Gnd of Feather Wing
 * SDA of Feather Wing
 * SCL of Feather Wing
*/
// Define Libraries
// ================
#include <Adafruit_SSD1306.h>
#include "Adafruit_GFX.h"
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <Wire.h>
#include <SPI.h>
#include <BlynkSimpleEsp8266.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <SimpleTimer.h>
#include <DHT.h>

// Glaobal Variables
// -----------------
int PI_Out;              // Custom Control Function Return Value
const double delta_time = 1.2;  // 1.2 Second Sample Rate in Auto (glaobal variable)
double I_Term = 0.0;       // Integral Term (global variable)
double output = 0.0;
const double  windup_guard = 500.0;  // Integral Winup prevention
double error = 0.0;
```

```
double Kp = 780.0;          // Proportiional Gain (global variable)
double Ki = 50.0;          // Integral Gain (global variable)
double setpoint = 70.0;    // Setpoint (global variable)
double alarm = 50.0;       // alarm setpoint (global variable)
double Tf;  // Global Temperature storage variable for Deg F
double h;  // Global Humidity variable
double t; // Global temp Deg.C
#define button_B 16  // This is the "B" button for manual operation
boolean flag = 0;  // Manual Operation Flag

// DHT Sensor Characteristics
// --------------------------
#define DHTPIN 12 //pin gpio 12 in sensor
#define DHTTYPE DHT22   // DHT 22 Change this if you have a DHT11
DHT dht(DHTPIN, DHTTYPE);

// Motor Drive Pin:
// ================
#define PWM_Pin 13 // Motor Drive Pin

// OLED Screen Setup
// -----------------
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Declare Global Variables
// ========================
char auth[] = "[redacted]";  // Put your Auth Token here.

SimpleTimer timer; // Create a Timer case

BLYNK_WRITE(V2){  // Read the slider (set up for 50-80 on display)
  setpoint = param.asFloat(); // Create a local variable to store virtual slider
  value
}

BLYNK_WRITE(V3){  // Read the slider (set up for 1-100 on display)
  Kp = param.asFloat(); // Create a local variable to store virtual slider value
}

BLYNK_WRITE(V4){  // Read the slider (set up for 0-50 on display)
  Ki = param.asFloat(); // Create a local variable to store virtual slider value
}

BLYNK_WRITE(V5){  // Read the slider (set up for 20-70 on display)
  alarm = param.asFloat(); // Create a local variable to store virtual slider value
}

// Main Program:
// =============
void setup() {
  pinMode(DHTPIN, INPUT_PULLUP); // Use internal resistor pullup on DHT Sensor
  pinMode(PWM_Pin, OUTPUT); // PWM Channel
  pinMode(2, OUTPUT); // Set blue LED as an output to indicate satisfactory Wi-Fi
  connection
  digitalWrite(2, HIGH);  // Turn off blue LED
  pinMode(button_B, INPUT); // Set Button B as input
  Serial.begin(115200);
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has shut
  off
  display.clearDisplay();  // Clear buffer
```

```
127     display.setTextSize(1); // Set OLED text size ("1" provides 4 lines of data about 20
        characters per line)
128     // ("2" provides 2 lines of data about 10 characters per line)
129     display.setTextColor(WHITE);  // Sets color
130     WiFiManager wifi;  // Start an AP, and a GUI to enter Wi-Fi information if one
        already does not exist
131     wifi.setAPCallback(configModeCallback);  // Goto Function if a pre-assigned WiFi
        credentials do not exist
132     wifi.autoConnect("Jon Humidor");
133     Blynk.config(auth); // Make the cloud server connection
134     dht.begin(); // Initialize DHT sensor
135     delay(1000); // Allow board to settle, and enough time to enter "boot mode"
136     if (WiFi.status() == WL_CONNECT_FAILED) {  // Displat banner if WiFi connection failed
137       digitalWrite(2, HIGH);   // Turn off blue LED (Wifi)
138       Serial.println("WiFi Connection Failed");  // Debug
139       display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has
          shut off
140       display.clearDisplay();  // Clear buffer
141       display.setTextSize(1); // Set OLED text size ("1" provides 4 lines of data about
          20 characters per line)
142        // ("2" provides 2 lines of data about 10 characters per line)
143       display.setCursor(0,0); // Set cursor to line 1, position zero, use (0,16) to go to
          second line, etc.
144       display.println("WiFI Connection");
145       display.println();
146       display.println("Has Failed");
147       display.display();  // Write to Display Buffer
148       delay(2000); // 2 second delay
149      }
150     Serial.println("Connected to WiFi");  // Debug
151     digitalWrite(2, LOW);  // Turn on BLUE LED to show Wi-Fi connection
152     display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has shut
        off
153     display.clearDisplay();  // Clear buffer
154     display.setTextSize(1); // Set OLED text size ("1" provides 4 lines of data about 20
        characters per line)
155     // ("2" provides 2 lines of data about 10 characters per line)
156     display.setCursor(0,0); // Set cursor to line 1, position zero, use (0,16) to go to
        second line, etc.
157     display.println("WiFI Connected To:");
158     display.println(WiFi.SSID());
159     display.println("IP Address: ");
160     display.println(WiFi.localIP()); // Display IP Address
161     display.display();  // Write to Display Buffer
162     delay(2000); // 2 second delay
163     display.clearDisplay();  // Clear buffer
164     display.setTextSize(1); // Set OLED text size ("1" provides 4 lines of data about 20
        characters per line)
165     // ("2" provides 2 lines of data about 10 characters per line)
166     display.setTextColor(WHITE);  // Sets color
167     display.setCursor(0,0); // Set cursor to line 1
168     display.println("     Jon's WiFI");  // Display Welcome message
169     display.println();
170     display.println("    Cigar Humidor");
171     display.display();  // Write to Display Buffer
172     timer.setInterval(1000L, SensorData); // Setup a function to be called every second
173     timer.setInterval(1200L, controller); // Setup a function to be called every 1.2
        seconds
174     timer.setInterval(2000L, senddata); // Setup a function to be called every  2 seconds
175     timer.setInterval(3000L, tempdisplay); // Setup a function to be called every 3 seconds
176     timer.setInterval(4000L, humdisplay); // Setup a function to be called every 4
        seconds
177     timer.setInterval(5000L, blankscreen); // Setup a function to be called every 5 seconds
178     timer.setInterval(600000L, sendNotifications); // Setup a function to be called every
        10 minutes
179     if (digitalRead(button_B) == 0){
180        flag = 1; // Set program flag to manual operation
181        ManOperation(); // Go to manual Operation Function
182     }
```

```
183     delay(2000); // 2 second delay
184   }
185
186   BLYNK_CONNECTED() {
187     Blynk.syncAll(); // Sync all virtual devices to last values stored on server.
188   }
189
190   void loop() {
191     Blynk.run();
192     timer.run();
193   }
194
195   // Screen Blanking Function:
196   // =========================
197   void blankscreen(){
198     display.clearDisplay();
199     display.ssd1306_command(SSD1306_DISPLAYOFF); // Switch display off
200   }
201
202   // PI Controller Function:
203   // =======================
204   float Calculate_PI () {
205       error = setpoint - h;  // Error Term, h = feedback
206       I_Term += (error * delta_time);  // Intergral Term
207       if (I_Term > windup_guard){ // Positive Integral Windup Guard
208          I_Term = windup_guard;
209       }
210       if (I_Term < -windup_guard){  // Negative Integral Windup Guard
211          I_Term = -windup_guard;
212       }
213       if (isnan(I_Term)){  // Reset if NAN
214         I_Term = 0;
215       }
216       output = (Kp * error) + (Ki * I_Term);   // Controller Output (Proportional +
      Integral)
217       if (output <= 0.0) {  // Limit output to positive PWM values only
218         output = 0.0;
219       }
220       if (output >= 1023.0) { //Limit output to positive PWM values only (ESP8266 uses 10
      bit resolution, so 1023)
221           output = 1023.0;
222       }
223      Serial.println("Kp = " + String(Kp));       // Debug
224      Serial.println("Ki = " + String(Ki));       // Debug
225      Serial.println("Setpoint = " + String(setpoint));  // Debug
226      Serial.println("Feedback (humidity) = " + String(h));  // Debug
227      Serial.println("Error = " + String(error));       // Debug
228      Serial.println("I_Term = " + String(I_Term));       // Debug
229      Serial.println("Ki *I_Term = " + String(Ki * I_Term));       // Debug
230      Serial.println("P_Term = " + String(Kp * error));  // Debug
231      Serial.println("Output = " + String(output));  //  Debug
232      Serial.println("Alarm Setpoint = " + String(alarm));  //  Debug
233      return int(output); // Return PI Control Value as an integer
234   }
235
236   // Send Data To Blynk Server Function
237   // ==================================
238   void senddata(){ //Read the Temp and Humidity from DHT, do not send more than 10 values
      per second
239     Blynk.virtualWrite(1, Tf); // virtual pin to display Temp
240     Blynk.virtualWrite(0, h); // virtual pin  to display Humidity
241     if (WiFi.status() == WL_CONNECTION_LOST) { // Checks for lost WiFi connection
242       digitalWrite(2, HIGH);  // Turn off Wifi LED (blue)
243       Serial.println("WiFi_CONNECTION_LOST");  // Debug
244       display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has
      shut off
245       display.clearDisplay();  // Clear OLED Display Buffer
246       display.setTextSize(1); // Set OLED Text Size
247       display.setTextColor(WHITE); // Set Color
```

```
248       display.setCursor(0,0);  // Set cursor position line 1
249       display.println("WiFi Connection"); // Write message
250       display.println();
251       display.println("Was Lost");
252       display.display(); // Write to OLED
253       delay(2000); // 2 second delay
254     }
255     else {
256       digitalWrite(2, LOW);  // Turn on Wifi LED (blue)
257     }
258   }
259
260   // Function to Send Notifications and E-Mails to Phone:
261   // ====================================================
262   void sendNotifications(){ // Send Notifications Function
263     if (h < alarm){ // Note: limitation is 1 notification and E-Mail per 15 seconds.
264       Blynk.notify("Humidor Humidity < Alarm Setpoint");
265       // Send e-mail when your hardware gets connected to Blynk Server
266       // Just put the recepient's "e-mail address", "Subject" and the "message body"
267       Blynk.email("u003rhg@gmail.com", "Low Humidity", "Humidor Humidty Is Less Than
          Alarm Setpoint");
268      }
269   }
270
271   // Function to Read Sensor:
272   // ========================
273   void SensorData() {  // Read DHT-22 Sensor
274       h = dht.readHumidity(); // Local Humidity storage variable
275       t = dht.readTemperature(); // Local Temperature storage variable Deg C
276       if (isnan(t) || isnan(h)){ // Compare temperature & humidity events and perform a
          check.
277       Serial.println("Bad Sensor Value");
278       //PI_Out = 0;  // turn off PWM, problem with sensor
279       display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has
          shut off
280       display.clearDisplay();  // Clear OLED Display Buffer
281       display.setTextSize(1); // Set OLED Text Size
282       display.setTextColor(WHITE); // Set Color
283       display.setCursor(0,0);  // Set cursor position line 1
284       display.println("Bad Check Sum"); // Write message
285       display.println();
286       display.println("Check Sensor");
287       display.display(); // Write to OLED
288       }
289       Tf = ((t * 9/5) + 32); // Convert temperature to degrees Fahrenheit
290       Serial.println("Humidity: ");
291       Serial.println(h);
292       Serial.println("Temperature: ");
293       Serial.println(Tf);
294       Serial.println("");
295   }
296
297   // Function to Display Temperature on OLED:
298   // ========================================
299   void tempdisplay(){ // Temperature display Function
300    display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has shut
      off
301    display.clearDisplay();  // Clear OLED Display Buffer
302    display.setTextSize(2); // Set OLED Text Size Larger than introduction banner
303    display.setTextColor(WHITE); // Set Color
304    display.setCursor(0,0);  // Set cursor position line 1
305    display.println("Temp_DegF:"); // Write Banner
306    display.setCursor(10,16);  // Set cursor position line 2 with text size = 2, and
      indent 5
307    display.println(Tf);  // Write Temperature
308     //display.println(" Deg. F"); // uncomment to append next to to float value
309    display.display(); // Write to OLED
310   }
311
```

```
312  // Function to Display Humidity on OLED:
313  // ====================================
314  void humdisplay(){ // humidity display Function
315   display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has shut
     off
316   display.clearDisplay();  // Clear OLED Display Buffer
317   display.setTextSize(2); // Set OLED Text Size Larger than introduction banner
318   display.setTextColor(WHITE); // Set Color
319   display.setCursor(0,0);  // Set cursor position line 1
320   display.println("Humid_%RH:"); // Write Banner
321   display.setCursor(10,16);  // Set cursor position line 2 with text size = 2, and
     indent 5
322   display.println(h); // Write Humidity
323    //display.println(" % RH"); // uncomment to append next to to float value
324   display.display(); // Write to OLED
325  }
326
327  // Function to Read Control Loop and set PWM:
328  // ==========================================
329  void controller(){ // Controller Function  Block
330    PI_Out = Calculate_PI(); // Calculate new PI Control Value
331    Serial.println("PI_Out = " + PI_Out);
332    analogWrite(PWM_Pin, PI_Out);  // PWM Value (0-1023)
333  }
334
335  // Function for callback on WiFi Startup:
336  // --------------------------------------
337  void configModeCallback (WiFiManager *myWiFiManager) {
338    Serial.println("Entered config mode");
339    digitalWrite(2, HIGH);  // Turn off blue LED
340    display.setCursor(0,0); // Set cursor to line 1, position zero, use (0,16) to go to
     second line, etc.
341    display.println("Starting Access Point");  // Display Message
342    display.println("Establish Network");  // Display Message
343    display.println("To > Jon Humidor");  // Display Message
344    display.println("Use Phone or Computer"); // Display Message
345    display.display();  // Write to Display Buffer
346    delay(1000);  // 1 second delay
347    Serial.println(".....................");
348  }
349
350  // Function for Manual Operation:
351  // ------------------------------
352  void ManOperation(){ //Read the Temp and Humidity from DHT
353    WiFiManager wifi;  // Start an AP, and a GUI to enter Wi-Fi information if one
     already does not exist
354    wifi.resetSettings(); // Reset wi-fi settings
355    digitalWrite(2, HIGH);  // Turn off blue LED
356    while (flag == 1){ // Manual operation flag
357      display.ssd1306_command(SSD1306_DISPLAYON); // Switch display back on, if it has
       shut off
358      display.clearDisplay();  // Clear buffer
359      display.setTextSize(1); // Set OLED text size ("1" provides 4 lines of data about
       20 characters per line)
360       // ("2" provides 2 lines of data about 10 characters per line)
361      display.setTextColor(WHITE);  // Sets color
362      display.setCursor(0,0); // Set cursor to line 1, position zero, use (0,16) to go to
       second line, etc.
363      display.println("Manual Mode of");  // Display Welcome message
364      display.println("Operation");
365      display.println("Press Reset to");
366      display.println("Return");
367      display.display();  // Write to Display Buffer
368      delay(1000); // 1 second delay
369      SensorData();  // Goto Function to read sensor data
370      delay(1000); // 1 second delay
371      humdisplay();  //  Goto Function to display Humidity
372      delay(1000); // 1 second delay
373      tempdisplay();  // Goto Function to display Temperature
```

```
      delay(1000); // 1 second delay
      Kp = 780.0;  // Proportiional Gain (fixed)
      Ki = 50.0;  // Integral Gain (fixed)
      setpoint = 70.0;  // Setpoint (fixed)
      controller();  // Goto Function to set PWM
      blankscreen(); // Goto Function to blank screen
      delay(1000); // 1 second delay
    }
  }
```