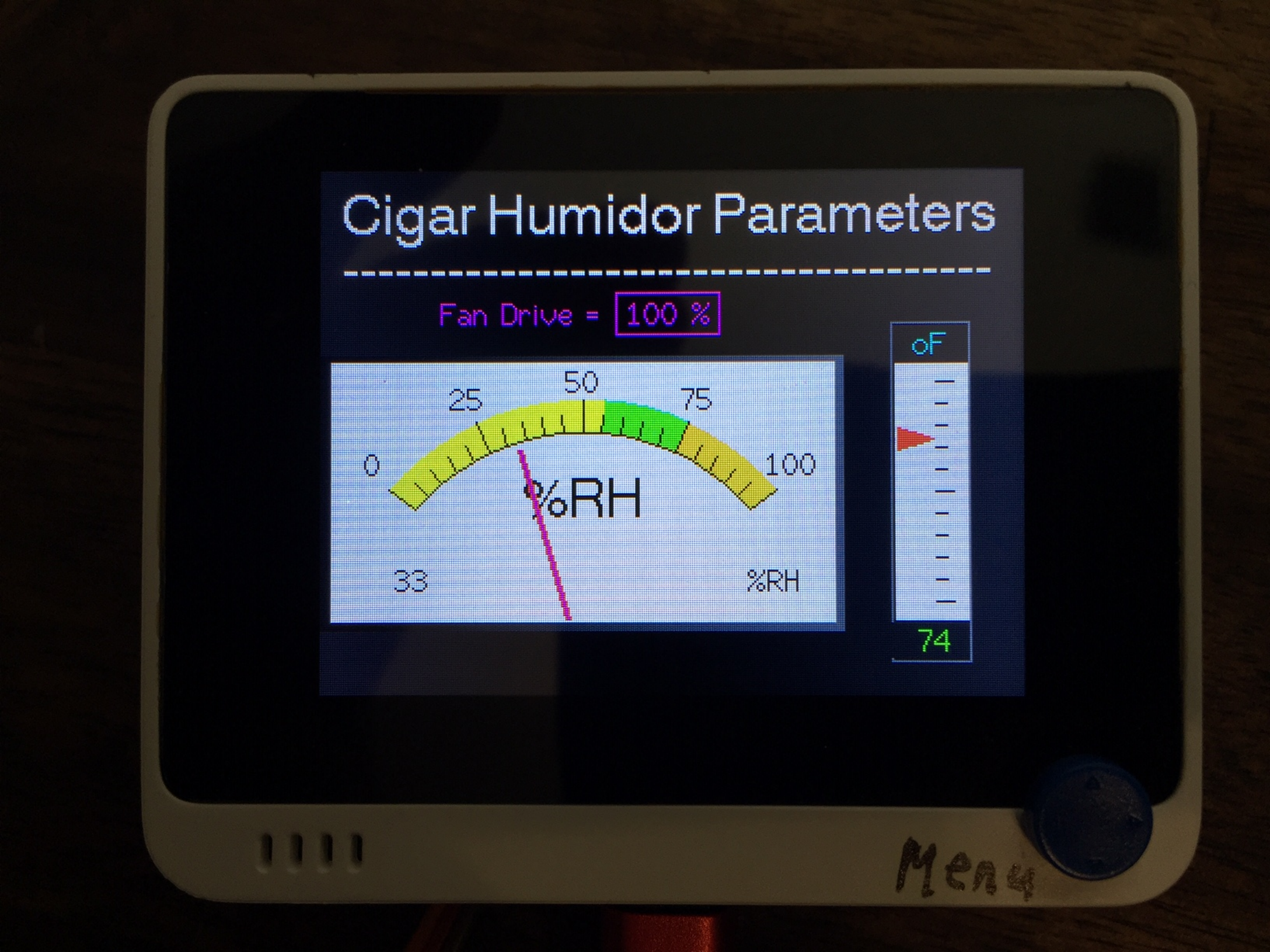

Cigar Humidor Parameters
Fan Drive = 100 %
0
25
50
75
100
%RH
33
%RH
oF
74
Menu

To Build the circuit follow the following:

WiFi Overview

Update the Wireless Core Firmware

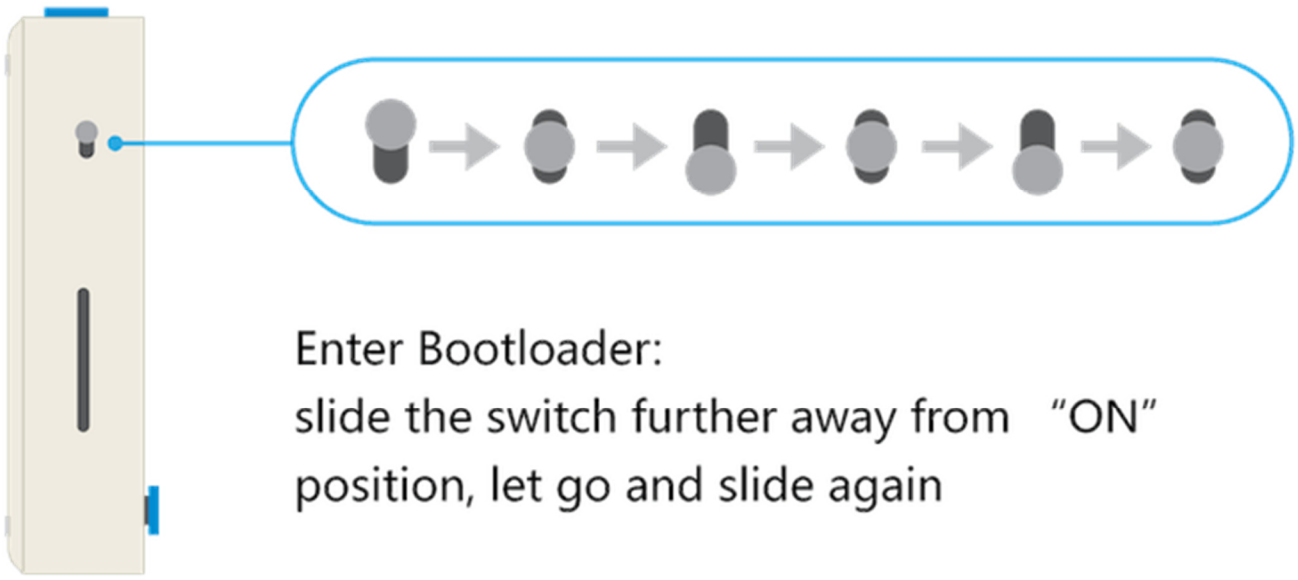
First, You need to update the firmware for the Realtek RTL8720 Wireless core on Wio Terminal.

Step 1 - Arduino Configuration

To be able to update the firmware on the RTL8720, we need to enable the Serial connection from SAMD51 to RTL8720. Seeed provides `uf2` methods of uploading Wio Terminal's firmware. Simply download the `uf2` files from below.

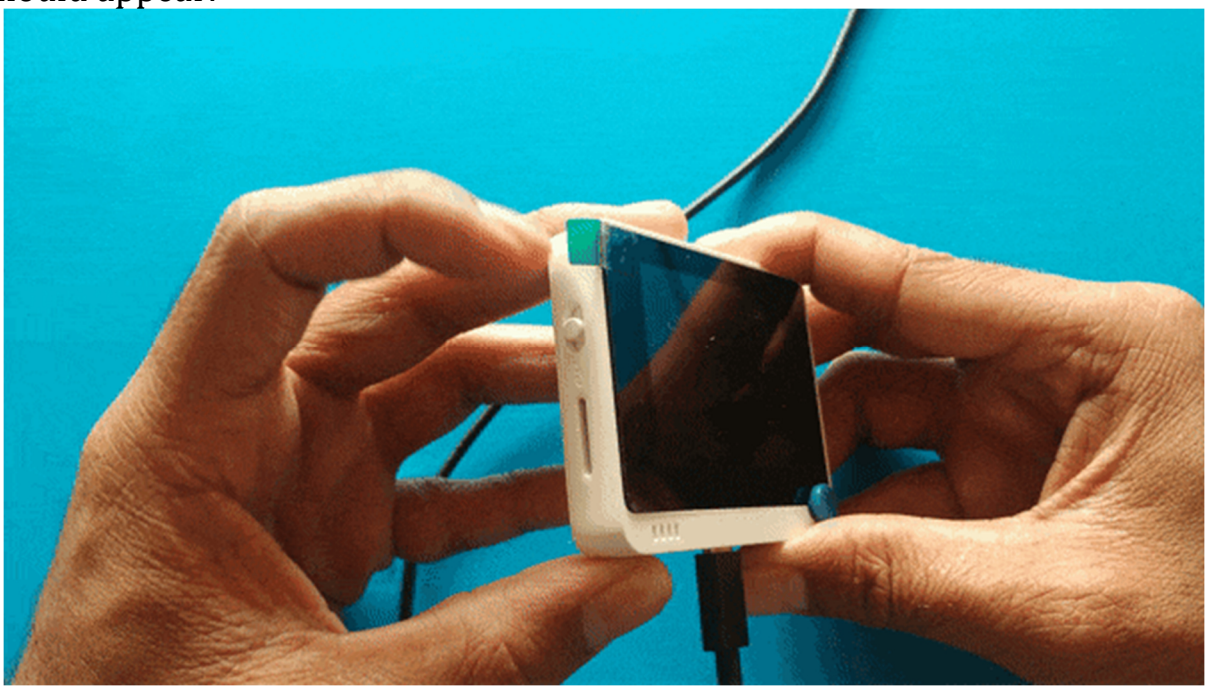
- Download the rtl8720_update_v2.uf2`uf2` files.

Step 1:1 Entering the bootloader mode by sliding the power switch twice quickly.
To Enter Bootloader: Slide the switch twice very quickly, as followed:



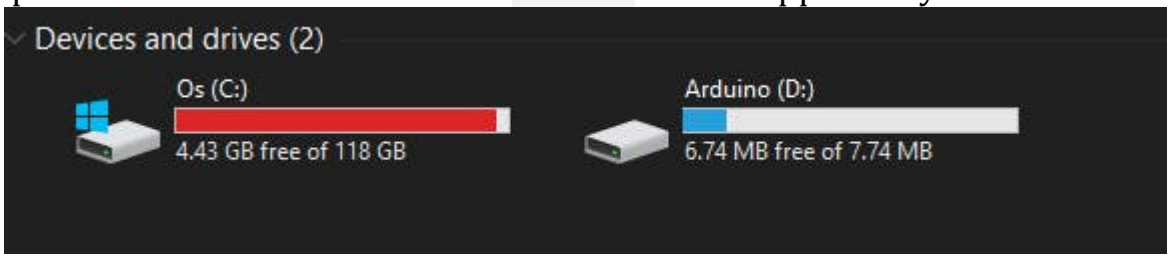
To Eneter Bootloader Mode

Once Wio Terminal is in the Bootloader mode, the blue LED will start to breathe in a way that is different from blinking. Check the port again and it should appear.



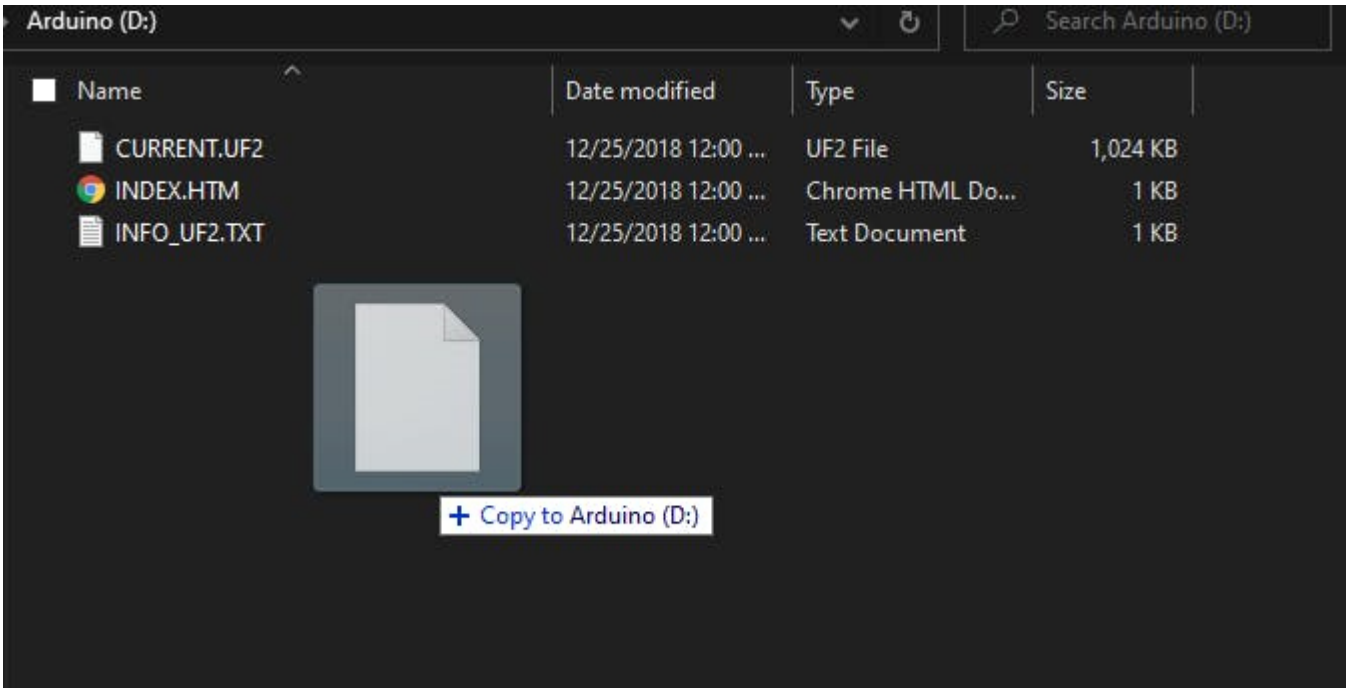
Bootloader Mode

Step 1.2: An external drive named `Arduino` should appear in your PC.



Arduino Drive

Drag the downloaded `rtl8720_update_v2.uf2` files into the `Arduino` drive and it will reset the Wio Terminal and loaded the sketch!



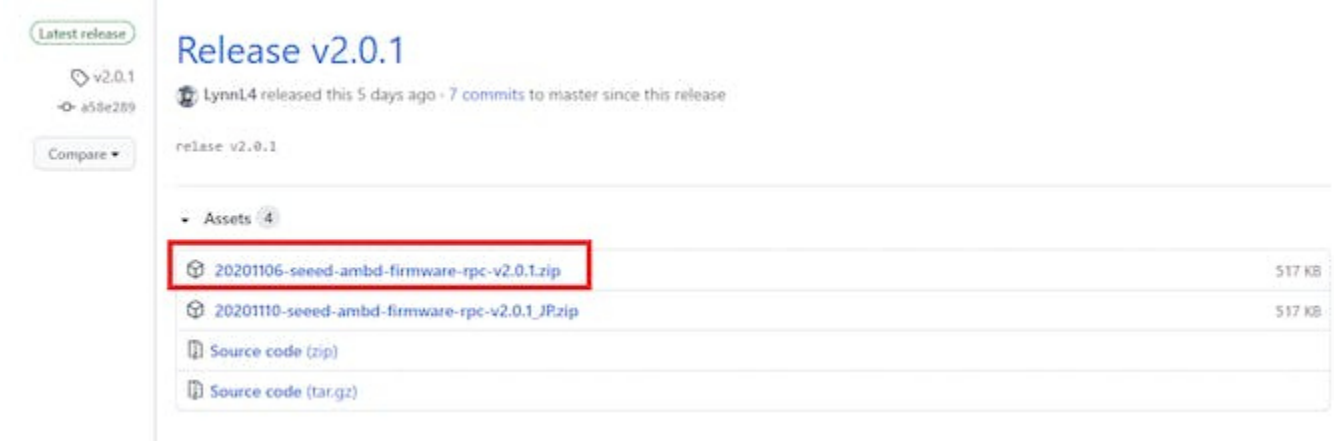
drag and drop the .uf2 files in to arduino drive

After that, you should see that Burn RTL8720 fw on the Wio Terminal's screen. This means that it is currently in the burning firmware mode!

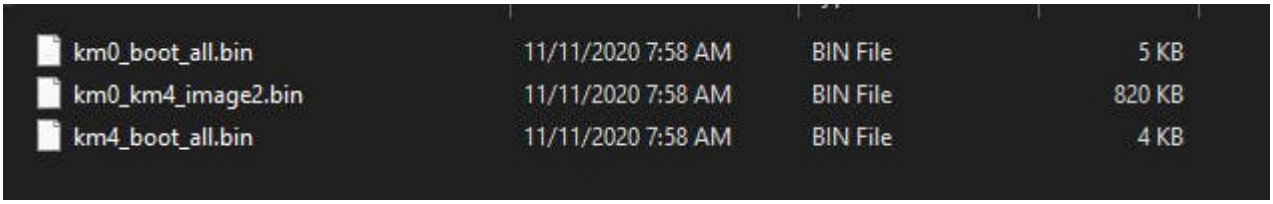
Step 2 - Download the Latest Firmware

You can download the latest eRPC Structure Firmware for RTL8720

- Download the latest RTL8720 FirmwareHere.



Note that the version might change in future.



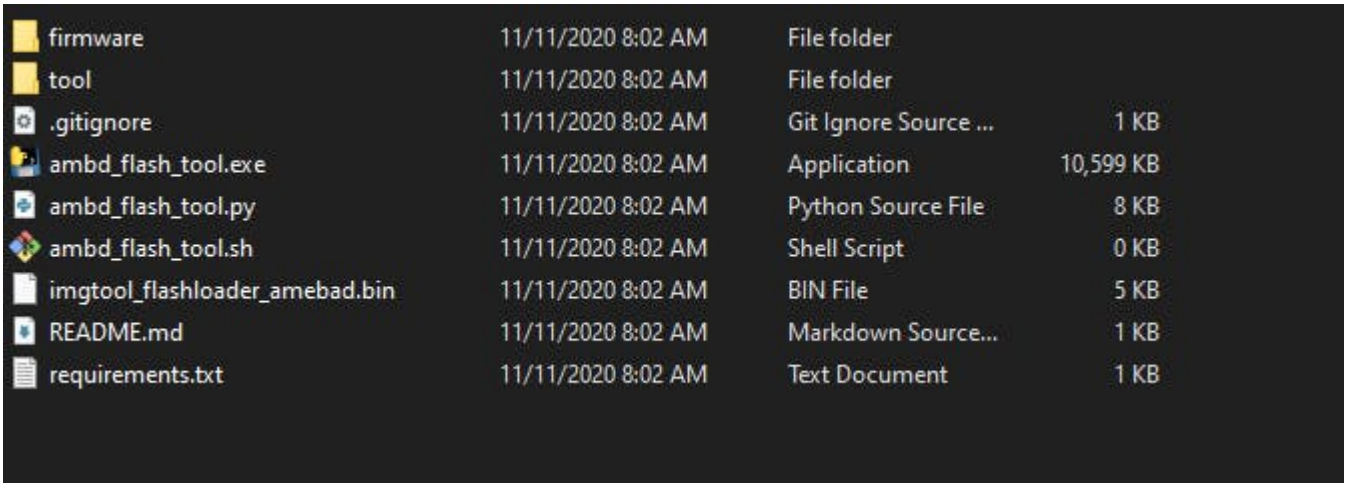
Firmware binary

Step 3 - Download Flash Tool

Next, you can download the flash tool.

Goto LynnL4/ambd_flash_tool and download the whole repo by clicking download ZIP or simply click here

Unzip the file and you can see the tool



Flash tools

After downloading the tools you can flash the RTL8720 firmware to Wio Terminal using the CLI methods.

- For macOS and LinuxOS, please use the `ambd_flash_tool.py` script.
- For Windows OS, please use the `ambd_flash_tool.exe` script.

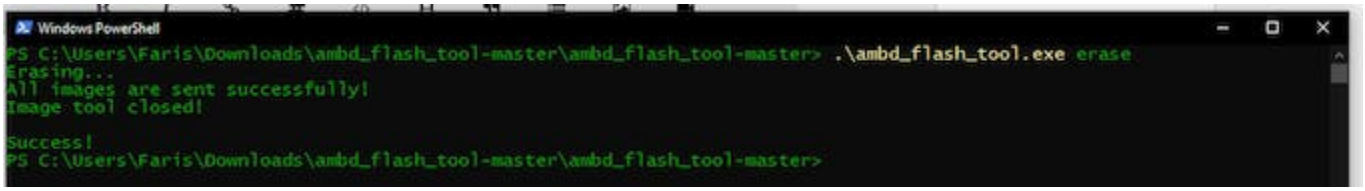
Since I was using the windows, I'll go with the `ambd_flash_tool.exe` to flash the firmware on wio terminal.

Note – Highlight the ambd_flash-tool.exe file, and then go to the “file” heading on folder, and click the option to run in windows powershell as administrator.

Step 4 - Erase Initial Firmware

First, we need to erase initial firmware inside the RTL8720, for that run:

Open the flash tool folder and open the PowerShell from the directory or you can open PowerShell and navigate to the directory.



erase

To Erase

```
.\ambd_flash_tool.exe erase
```

note that, it will take about 3 minutes some times to complete the erasing process, so please wait until you get the success message.

Step 5 - Flash New Firmware

Note – I placed all “3” bin files in a folder called “New_Firm” located on my Desktop

To flash the newly downloaded firmware into the RTL8720, run:

```
.\ambd_flash_tool.exe flash -d [RTL8720-firmware-path]
```

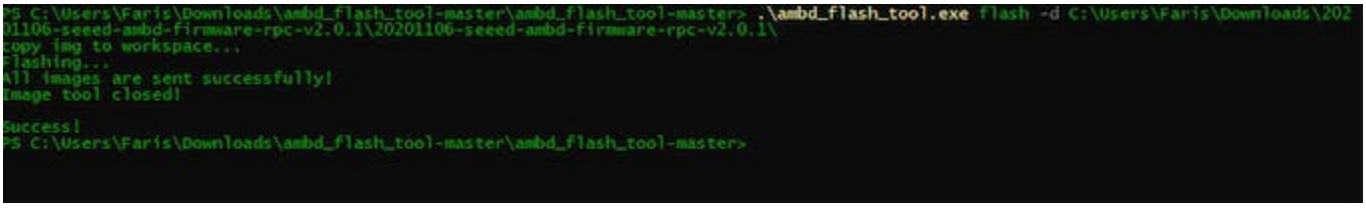
Note – For this next step ensure that Arduino is open and that you have conne cted to the “Com Port”

For it's on the download folder and I need to mention the full path.

```
.\ambd_flash_tool.exe flash -d C:\Users\u003r\Desktop\New_Flas
h
```

Flash

Please wait until you get the success message



Great, Flashing Completed 🔥. If you facing any issues while flashing, post your queries at SeeedStudio Forum

Installing Libraries

As part of the ePRC Firmware, seeed provided few libraries that are needed for the wireless connectivity.

- Seeed_Arduino_rpcBLE
- Seeed_Arduino_rpcWiFi
- Seeed_Arduino_FreeRTOS

The rpcWiFi software library calls Seeed Arduino rpcUnified to implement WiFi and BLE function compatibility with Arduino-ESP32. To reduce the cost of using the software, you can import your favourite ESP32 wifi app and BLE app directly, with minor changes, and then use it. You'll find that your favourite ESP32 app has 5G features and has BLE5.0 features, runs on ARM and other architectures.

1. Install the Seeed_Arduino_rpcWiFi

Visit the Seeed_Arduino_rpcWiFirepositories and download the entire repo to your local drive.

- Visit the Seeed_Arduino_rpcWiFirepositories and download the entire repo to your local drive.
- Now, the Seeed_Arduino_rpcWiFi library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_rpcWiFi` file that you have just downloaded.

2. Install the Seeed_Arduino_rpcBLE

Visit the Seeed_Arduino_rpcBLE repositories and download the entire repo to your local drive.

- Visit the Seeed_Arduino_rpcBLE repositories and download the entire repo to your local drive.
- Now, the Seeed_Arduino_rpcWiFi library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_rpcBLE` file that you have just downloaded.

3. Install the Seeed_Arduino_rpcUnified

Visit the Seeed_Arduino_rpcUnifiedrepositories and download the entire repo to your local drive.

- Visit the Seeed_Arduino_rpcUnifiedrepositories and download the entire repo to your local drive.
- Now, the Seeed-Arduino-FreeRTOS library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_rpcUnified` file that you have just downloaded

4. Install the Seeed_Arduino_FreeRTOS ¶

Visit the Seeed_Arduino_FreeRTOS repositories and download the entire repo to your local drive.

- Visit the Seeed_Arduino_FreeRTOSrepositories and download the entire repo to your local drive.
- Now, the Seeed-Arduino-FreeRTOS library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_FreeRTOS` file that you have just downloaded.

5. Install the File System Library

- Visit the Seeed_Arduino_FS repositories and download the entire repo to your local drive.
- Now, the FS library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_FS` file that you have just downloaded.

Installing the Dependent SFUD Libraries

- Visit the Seeed_Arduino_SFUD repositories and download the entire repo to your local drive.
- Now, the SFUD library can be installed to the Arduino IDE. Open the Arduino IDE, and click `sketch` -> `Include Library` -> `Add .ZIP Library`, and choose the `Seeed_Arduino_SFUD` file that you have just downloaded.

6. Install the Seeed_Arduino_mbedtls - search for "seeed mbedtls" under libraries

After installing all the required libraries, you are all set to do some BLE and WiFi Hacks .

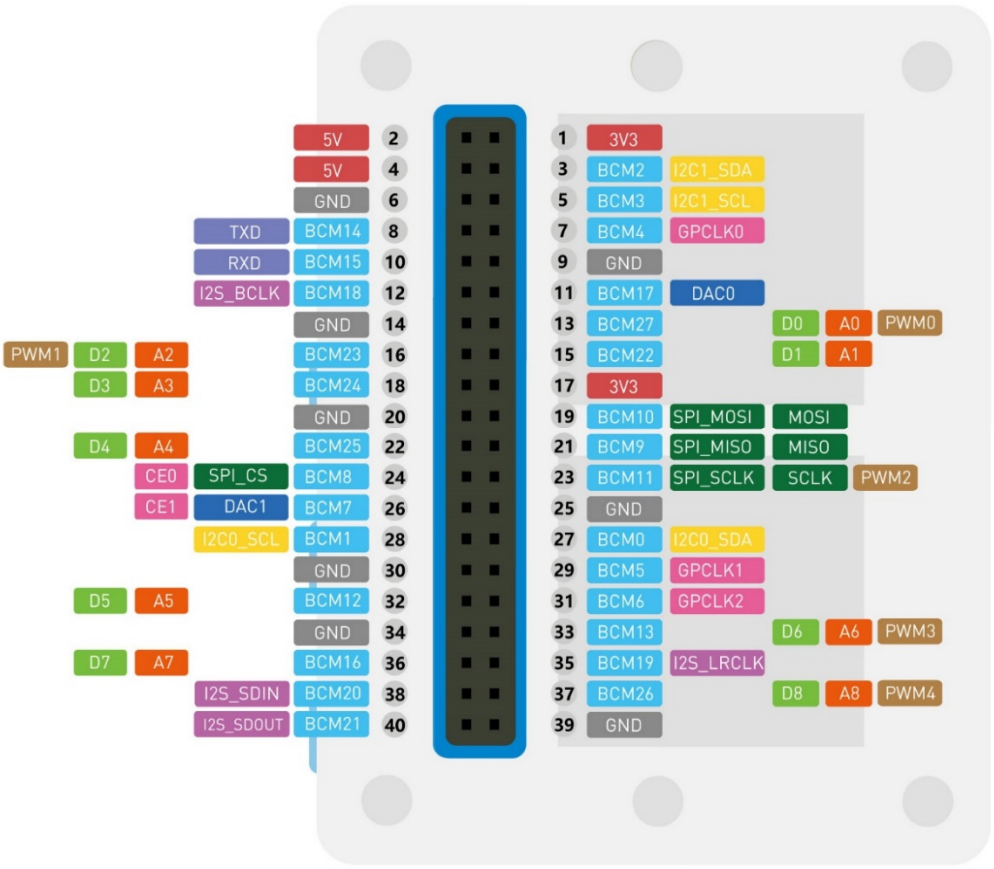
I tried to scan both available WiFi access point and Bluetooth devices together, and it works like a charm

7. Install the following additional support libraries available on “GITHUB” using the “ZiP” install method above:

- **FlashStorage_SAMD.h // Used to store EEPROM Settings from Menu**
- **DNSServer.h**
- **WebServer.h**
- **WiFiManager.h // Seed Studio version**
- **DHT.h // Groove DHT Temperature and Humidity library**

Note – if you get a compiling error be sure to look at the “include statements” which are the libraries that were installed, to ensure that you are not missing any!

Hardware Pinout Quick Overview:



Code Description (Shown at the end of this document):

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

3) Auto reconnects Wifi if it is dropped

4) Detects Sensor failure and displays an error message

5) Top Left Buttun Resets WiFi Settings (press and hold upon power-up only)

6) Contains a Built in WiFi Manager to connect Wio Device to your home router via a Graphical User Interface

7) 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

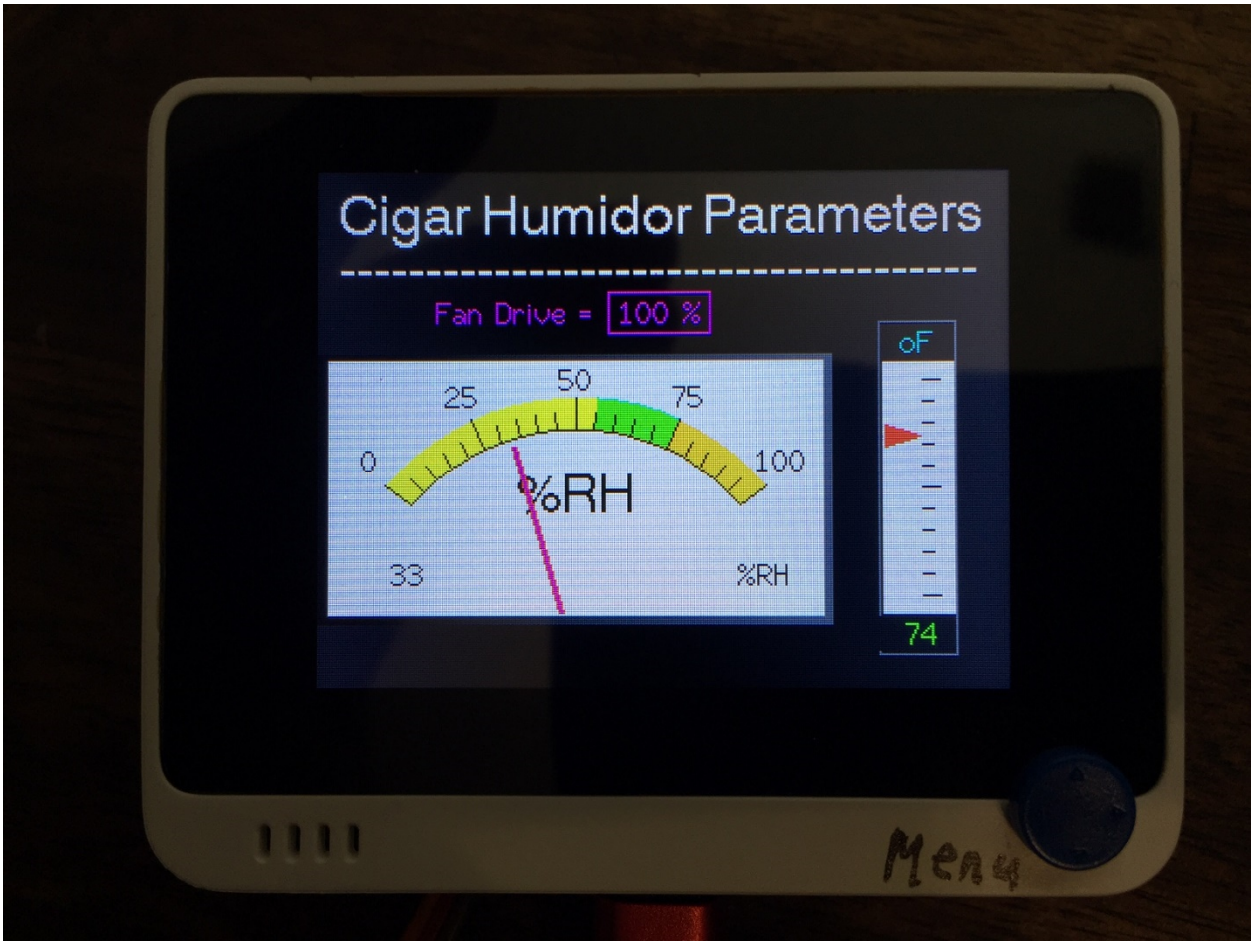
8) Fan Control is PWM controlled through pin "A8" and GND and +3.3v via a 2N3904 driver transistor (see schematic)

9) Parameters are stored in EEPROM and read into program once set, otherwise they start as “default”

10) Contains an internal buzzer when humidity falls below the user setpoint

11) WiFi settings are saved once set up. If you get a message to "open browser..." * cycle power and that uses fixes

Picture of Wio Terminal once completed code is uploaded, and everthing is functional:



```
/*
 * 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.
 * 3) Auto reconnects Wifi if it is dropped.
 * 4) Detects Sensor failure and displays an error message.
 * 5) Top Left Buttun Resets WiFi Settings upon power-up only.
 * 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.
 * 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.
 * 12) WiFi settings are saved once set up. If you get a message to "open browser..."
 *     cycle power and that uses fixes.
 */

// Libraries:
// ----------
#include <FlashStorage_SAMD.h>
#include <rpcWiFi.h>
#include <DNSServer.h>
#include <WebServer.h>
#include <WiFiManager.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;
```

```
70  const unsigned long period1 = 6000;  //the value is a number of milliseconds (6 seconds)
71  #define FLASH_DEBUG  0
72  #define TFT_GREY 0x5AEB
73  #define LOOP_PERIOD 35 // Display updates every 35 ms
74  float ltx = 0;    // Saved x coord of bottom of needle
75  uint16_t osx = 120, osy = 120; // Saved x & y coords (osx = 120, osy = 120)
76  uint32_t updateTime = 0;       // time for next update
77  int old_analog =  -999; // Value last displayed
78  int old_digital = -999; // Value last displayed
79  int value[6] = {0, 0, 0, 0, 0, 0};
80  int old_value[6] = { -1, -1, -1, -1, -1, -1};
81  int d = 0;
82  boolean interlock = true; // Stops Program execution while in Menu
83
84  // DHT Sensor Characteristics (Uncomment whatever type you're using)
85  // ------------------------------------------------------------------
86  //#define DHTTYPE DHT11   // DHT 11
87  #define DHTTYPE DHT22   // DHT 22  (AM2302)
88  //#define DHTTYPE DHT21   // DHT 21 (AM2301)
89  //#define DHTTYPE DHT10   // DHT 10
90  //#define DHTTYPE DHT20   // DHT 20
91  #define DHTPIN D1     // Data Pin we're connected to
92  DHT dht(DHTPIN, DHTTYPE);   //   DHT11 DHT21 DHT22
93  //DHT dht(DHTTYPE);         //   DHT10 DHT20 don't need to define Pin
94
95  // Motor Drive Pin:
96  // ----------------
97  #define PWM_Pin A8 // Motor Drive Pin
98
99  WebServer server(80);  // Create Server on Port 80
100
101 // Main Program:
102 // =============
103
104 void setup() {
105   Serial.begin(115200);
106   tft.init();
107   tft.setRotation(3);
108   //tft.setTextSize(2);
109   tft.fillScreen(TFT_BLACK);
110   tft.setTextColor(TFT_WHITE);
111   tft.drawString("Cigar Humidor Controller", 10, 10, 4); //prints strings from (x, y,
      font size)
112   tft.drawString("With Advanced Features", 10, 50, 4);
113   tft.drawString("By; Roy H Guerra Jr.", 10, 90, 4);
114   pinMode(WIO_5S_UP, INPUT_PULLUP);  // Enable Wio Button puulup Resistors
115   pinMode(WIO_5S_DOWN, INPUT_PULLUP);
116   pinMode(WIO_5S_LEFT, INPUT_PULLUP);
117   pinMode(WIO_5S_RIGHT, INPUT_PULLUP);
118   pinMode(WIO_5S_PRESS, INPUT_PULLUP);
119   pinMode(WIO_KEY_A, INPUT_PULLUP);
120   pinMode(WIO_KEY_B, INPUT_PULLUP);
121   pinMode(WIO_KEY_C, INPUT_PULLUP);
122   pinMode(PWM_Pin, OUTPUT); // PWM Channel
123   pinMode(WIO_BUZZER, OUTPUT); // Internal Wio Buzzer
124   dht.begin(); // Initialize DHT sensor
125   delay(2000);  // 2S loop delay
126   tft.fillScreen(TFT_BLACK);
127   WiFiManager wifiManager;
128     if (digitalRead(WIO_KEY_C) == LOW) {
129     Serial.println("WiFi Reset");
130     wifiManager.resetSettings();
131     tft.fillScreen(TFT_BLACK); // Clear Screen
132     tft.setTextColor(TFT_RED);
133     tft.drawString("WiFi Settings Are Reset", 10, 30, 4); //prints strings from (x, y,
      font size)
134     tft.drawString("Turn Off Power Button", 10, 66, 4);
135     tft.drawString("Re-Start The Wio Device", 10, 102, 4);
136     }
```

```
    //delay(2000);  // Delay 2 seconds
      //set callback that gets called when connecting to previous WiFi fails, and enters Access Point mode
      wifiManager.setAPCallback(configModeCallback);
      //Fetches ssid and pass from RTL8720 and tries to connect
      //if it does not connect it starts an access point with the specified name
      //here  "AutoConnectAP"
      //and goes into a blocking loop awaiting configuration
      // delay(2000);  // Delay 2 seconds
      wifiManager.autoConnect("Wio Humidor");
      //if you get here you have connected to the WiFi
      Serial.println("WiFi Is Connected");
      Serial.println("IP Address = ");
      Serial.println(WiFi.localIP());
      Serial.println("SSID = ");
      Serial.println(WiFi.SSID());
      long rssi = WiFi.RSSI();
      Serial.println("RSSI = ");
      Serial.println(WiFi.RSSI());
      tft.fillScreen(TFT_BLACK); // Clear Screen
      tft.setTextColor(TFT_YELLOW);
      tft.drawString("Wifi Connected", 10, 30, 4); //prints strings from (x, y, font size)
      tft.setTextColor(TFT_CYAN);
      tft.drawString("SSID = " + String(WiFi.SSID()), 10, 70, 4);
      tft.setTextColor(TFT_MAGENTA);
      tft.drawString("IP Add = " + String(WiFi.localIP().toString()), 10, 110, 4);
      tft.setTextColor(TFT_BLUE);
      tft.drawString("RSSI = " + String(rssi) + " dBm", 10, 150, 4);
      delay(5000);  // 5 second Delay
      tft.fillScreen(TFT_BLACK); // Clear Screen
    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 Configuration (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));
```

```
271           }
272           if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 2)) {
273              Serial.println("A Key pressed");
274              if (Sp < 80){
275                Sp += 1;
276                tft.fillRect(245,150,55,35,TFT_BLACK);
277                tft.drawRect(245,150,55,35,TFT_WHITE);
278                tft.setTextColor(TFT_WHITE);
279                tft.drawString(String(Sp), 250, 160, 4);
280               }
281            Serial.println("Ha = " + String(Sp));
282            }
283           if (((digitalRead(WIO_KEY_C) == LOW)) && (count == 2)) {
284             Serial.println("C Key pressed");
285             /***
286             The function EEPROM.update(address, val) is equivalent to the following:
287             if( EEPROM.read(address) != val ) {
288              EEPROM.write(address, val);
289             }
290              ***/
291              EEPROM.update(address + 30, Sp);
292              EEPROM.commit();
293              Serial.println("Wrote " + String(Sp) + " To EEPROM  Address 130");
294              tft.fillScreen(TFT_BLACK);
295              tft.setTextColor(TFT_MAGENTA);
296              tft.drawString("Controller Setpoint Saved", 10, 70, 4);
297              delay(2000);  // 2S loop delay
298              tft.fillScreen(TFT_BLACK);
299              flag = 0;  // Reset flag
300              count = 3; // Next menu Option
301            }
302        delay(150);  // 150mS loop delay
303        }
304        break;
305       case 3:
306        tft.fillScreen(TFT_BLACK);
307        tft.setTextColor(TFT_GREEN);
308        tft.drawString("Set Proportional Gain", 10, 10, 4); //prints strings from (x, y,
       font size)
309        tft.drawString("------------------------------", 10, 30, 4);
310        tft.setTextColor(TFT_YELLOW);
311        tft.drawString("Press Top Right Button (+)", 10, 70, 4);
312        tft.drawString("Press Top Mid. Button (-)", 10, 110, 4);
313        tft.setTextColor(TFT_WHITE);
314        tft.drawString("Proportional Gain = ", 10, 160, 4);
315        tft.drawRect(245,150,55,35,TFT_WHITE);
316        tft.drawString(String(Kp), 250, 160, 4);
317        tft.setTextColor(TFT_RED);
318        tft.drawString("Press Top Left Button To", 10, 192, 4);
319        tft.drawString("Save Configuration (exit)", 10, 215, 4);
320        flag = 1; // Change program flag
321        while (flag == 1) {
322          if (((digitalRead(WIO_KEY_B) == LOW)) && (count == 3)){
323            Serial.println("B Key pressed");
324            if (Kp > 1){
325               Kp -= 1;
326               tft.fillRect(245,150,55,35,TFT_BLACK);
327               tft.drawRect(245,150,55,35,TFT_WHITE);
328               tft.setTextColor(TFT_WHITE);
329               tft.drawString(String(Kp), 250, 160, 4);
330             }
331            Serial.println("Kp = " + String(Kp));
332            }
333           if (((digitalRead(WIO_KEY_A) == LOW)) && (count == 3)) {
334              Serial.println("A Key pressed");
335              if (Kp < 100){
336                Kp += 1;
337                tft.fillRect(245,150,55,35,TFT_BLACK);
338                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)
       if (WiFi.status() != WL_CONNECTED) {   // reconnect WiFi if it gets dropped
       automatically
          WiFi.reconnect();
       }
       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));
```

```
474       Sp = EEPROM.read(address + 30);
475       Serial.println("Sp = " + String(Sp));
476       if ((Kp == 255) || (Ki == 255) || (Sp == 255) || (Ha ==255)) { // Guards against
          EEPROM not being set
477         Kp = 50;
478         Ki = 5;
479         Ha = 50;
480         Sp = 70;
481       }
482       error = Sp - Hum; // Error Term, h = feedback
483     I_Term += (error * delta_time); // Intergral Term
484     if (I_Term > windup_guard){ // Positive Integral Windup Guard
485       I_Term = windup_guard;
486     }
487     if (I_Term < - windup_guard){ // Negative Integral Windup Guard
488       I_Term = - windup_guard;
489     }
490     if (isnan(I_Term)){ // Reset if NAN
491       I_Term = 0;
492     }
493    output = (Kp * error) + (Ki * I_Term); // Controller Output (Proportional + Integral)
494    output = constrain(output, 0, 255); // Limits Controller Range
495    Serial.println("Kp = " + String(Kp)); // Debug
496    Serial.println("Ki = " + String(Ki)); // Debug
497    Serial.println("Setpoint = " + String(Sp)); // Debug
498    Serial.println("Feedback (humidity) = " + String(Hum)); // Debug
499    Serial.println("Error = " + String(error)); // Debug
500    Serial.println("I_Term = " + String(I_Term)); // Debug
501    Serial.println("Ki *I_Term = " + String(Ki * I_Term)); // Debug
502    Serial.println("P_Term = " + String(Kp * error)); // Debug
503    Serial.println("Output = " + String(output)); // Debug
504    Serial.println("Alarm Setpoint = " + String(Ha)); // Debug
505    return int(output); // Return PI Control Value as an integer
506  }
507
508  // Function to Sound Buzzer:
509  // -------------------------
510  void buzzer(){ // Buzzer Function Block
511    if (Hum < Ha){
512      analogWrite(WIO_BUZZER, 128);
513      delay(1000);
514      analogWrite(WIO_BUZZER, 0);
515      delay(1000);
516     }
517  }
518
519  // Function to Read Control Loop and set PWM and Speed Indication:
520  // ---------------------------------------------------------------
521  void controller(){ // Controller Function Block
522    PI_Out = Calculate_PI(); // Calculate new PI Control Value
523    Serial.println("PI_Out = " + String(PI_Out)); // Debug
524    analogWrite(PWM_Pin, PI_Out); // PWM Value (0-255)
525    Fs = map(output, 0, 255, 0, 100); // Rescale controller output to % fan speed
526    Serial.println("Fan Speed = " + String(Fs)); // Debug
527  }
528
529  // Read Sensor Function
530  // --------------------
531  void SensorData(){
532    Hum = dht.readHumidity(); // Measure the humidity
533    Serial.println("Humidity = " + String(Hum));
534    TemperatureC = dht.readTemperature(); // Measure the temperature
535    TempF = ((TemperatureC * 9/5) + 32); // Convert temperature to degrees Fahrenheit
536    Serial.println("Temperature = " + String(TempF));
537    // Compare temperature & humidity events and perform a check sum.
538    if (isnan(TemperatureC) || isnan(Hum)){ // Print "0" for a bad reading
539      TempF = 0;
540      Hum = 0;
541      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
677         tft.drawLine(120 + 20 * ltx, 205, osx, osy + 82, TFT_MAGENTA);
678         tft.drawLine(120 + 20 * ltx + 1, 205, osx + 1, osy + 82, TFT_RED);
679         // Slow needle down slightly as it approaches new postion
680         if (abs(old_analog - value) < 10) {
681             ms_delay += ms_delay / 5;
682         }
683         // Wait before next update
684         delay(ms_delay);
685     }
686 }
687
688 // Draw a meter on the screen:
689 // ---------------------------
690 void plotLinear(char* label, int x, int y) {
691     int w = 36;
692     tft.drawRect(x, y, w, 155, TFT_GREY);
693     tft.fillRect(x + 2, y + 19, w - 3, 155 - 38, TFT_WHITE);
694     tft.setTextColor(TFT_CYAN, TFT_BLACK);
695     tft.drawCentreString(label, x + w / 2, y + 2, 2);
696     for (int i = 0; i < 110; i += 10) {
697         tft.drawFastHLine(x + 20, y + 27 + i, 6, TFT_BLACK);
698     }
699     for (int i = 0; i < 110; i += 50) {
700         tft.drawFastHLine(x + 20, y + 27 + i, 9, TFT_BLACK);
701     }
702     tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 - 5, TFT_RED);
703     tft.fillTriangle(x + 3, y + 127, x + 3 + 16, y + 127, x + 3, y + 127 + 5, TFT_RED);
704     tft.drawCentreString("---", x + w / 2, y + 155 - 18, 2);
705 }
706
707 // Adjust the vertical linear meter pointer positions:
708 // ---------------------------------------------------
709 void plotPointer(void) {
710     value[0] = int(TempF); // Assign TempF to Value.
711     int dy = 187;  // 187
712     byte pw = 16;  // 16
713     tft.setTextColor(TFT_GREEN, TFT_BLACK);
714     // Move the 6 pointers one pixel towards new value
715     for (int i = 0; i < 1; i++) { // i < 6
716         char buf[8]; dtostrf(value[i], 4, 0, buf); //dtostrf(value[i], 4, 0, buf)
717         tft.drawRightString(buf, i * 40 + 287, 207, 2); // Value display (x, y, font
            size)
718         int dx = 263 + 40 * i; // Red Pointer "X" position
719         if (value[i] < 0) {
720             value[i] = 0;    // Limit value to emulate needle end stops
721         }
722         if (value[i] > 100) {
723             value[i] = 100;
724         }
725         while (!(value[i] == old_value[i])) {
726             dy = 180 + 17 - old_value[i]; // Red Pointer "Y" position
727             if (old_value[i] > value[i]) {
728                 tft.drawLine(dx, dy - 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +
                    pw, dy, TFT_WHITE
729                 old_value[i]--;
730                 tft.drawLine(dx, dy + 6, dx + pw, dy + 1, TFT_RED); //dx, dy + 6, dx +
                    pw, dy + 1, TFT_RED
731             } else {
732                 tft.drawLine(dx, dy + 5, dx + pw, dy, TFT_WHITE); //dx, dy - 5, dx +
                    pw, dy, TFT_WHITE
733                 old_value[i]++;
734                 tft.drawLine(dx, dy - 6, dx + pw, dy - 1, TFT_RED); //dx, dy + 6, dx +
                    pw, dy + 1, TFT_RED
735             }
736         }
737     }
738 }
739
```

```
// WiFi CallBack Routine:
// ---------------------
void configModeCallback (WiFiManager *myWiFiManager) {
  Serial.println("Entered config mode");
  Serial.println(WiFi.softAPIP());
  //if you used auto generated SSID, print it
  Serial.println(myWiFiManager->getConfigPortalSSID());
  tft.fillScreen(TFT_BLACK); // Clear Screen
  tft.setTextColor(TFT_WHITE);
  tft.drawString("On a phone, Tablet or PC", 10, 10, 4); //prints strings from (x, y,
  font size)
  tft.drawString("Goto Wifi Settings", 10, 34, 4);
  tft.setTextColor(TFT_GREEN);
  tft.drawString("Connect to Wio Terminal", 10, 70, 4);
  tft.setTextColor(TFT_CYAN);
  tft.drawString("Enter in SSID & Passwword", 10, 104, 4);
  tft.drawString("In Graphical Interface", 10, 128, 4);
  tft.setTextColor(TFT_RED);
  tft.drawString("If no Graphical Interface", 10, 164, 4);
  tft.drawString("Type 192.168.1.1", 10, 190, 4);
  tft.drawString("Inside a WEB Browser", 10, 216, 4);
  delay(1000); // 1 second delay
}
```