Heat and Humidity in the Workshop and a Dewpoint Monitor

For those people with a detached workshop, it is always difficult to keep tools and machines protected from damp in the winter. If the workshop is poorly insulated this adds to the headache for not just the tools but also the operator. If the only heating available is a bottled gas stove the problems dramatically increase as this style of heating produce copious amounts of water vapour which readily condenses on cold surfaces.

Better insulation should be the first step to improve the situation. Next comes choice of heater and a dry source of heat is essential. Spurred on by my local modelling club having fitted a diesel hot air heater, I also fitted one of these. This is mounted external to the workshop and has a feed in/feed out pipe through the wall. It works incredibly well and rarely needs more than an hour to get things up to working temperature. For those interested there is a blog link on this below.

As a side suggestion I find that any bright metal surfaces that do not get touched very often greatly benefit from a wipe coat of a 50/50 mix of thinners and linseed oil. I picked this tip up from the clockmaker William Smith. You have to be careful of the fire risk of the residual bundled cloths coated in linseed oil as they can spontaneously combust delivering a very warm workshop for a short period of time.

Once you have done the basics of insulation, heating and surface protection, you perhaps move into more scientific methods. If you can monitor the temperature and the humidity in the workshop you can calculate the dew point when condensation will start to occur.

If you have internet connectivity you can use a range of tags from Wireless Sensor Tags to monitor temperature and humidity and from this calculate the dew point.

These tags are relatively low cost and give a good radio coverage area. They each report back to a central node on your home network and the information is then stored in the Cloud for logging and reporting back to you via a desktop or phone app. You can set parameters so the tags send you a message by email if measurements hit alarm limits. They also have a built-in movement detector so you can fit them to doors etc to check for unwanted visitors. I have these fitted around the house both in the workshop and also domestically checking the freezer temperature, hot water temperature etc.

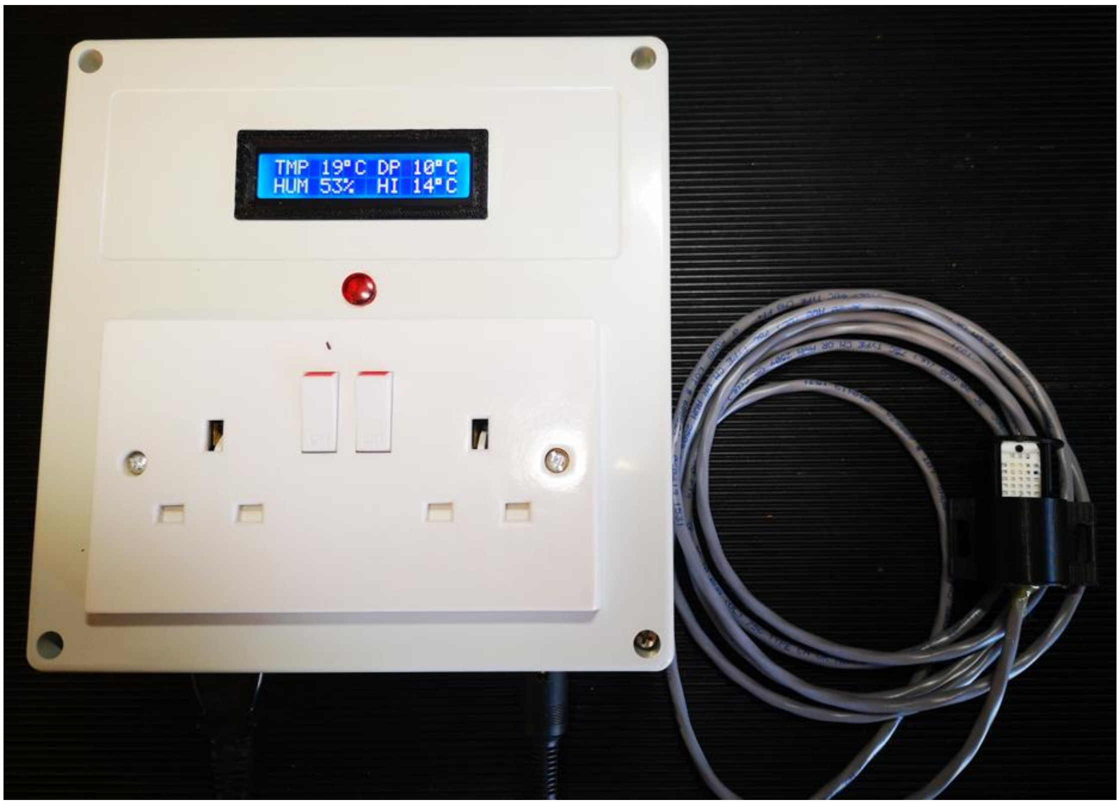
The problem with these particular tags (which I have yet to easily solve) is how to activate external devices such as heaters if certain temperature and humidity levels are reached. One alternative crude solution is to fit frost stats to trigger heaters but this seems overkill when the dew point might not have been reached.

The elegant techie solution is to build your own monitor that provides switched power to bring into play background heaters. There are a number of designs on the internet, many of which are based around Arduino controller modules. These are not difficult to replicate. By cherry picking from various internet published designs I have engineered a solution and details follow. It is not a difficult project and although I have created a PCB design for this, the wiring is sufficiently simple to allow a breadboard lash up to be used instead.

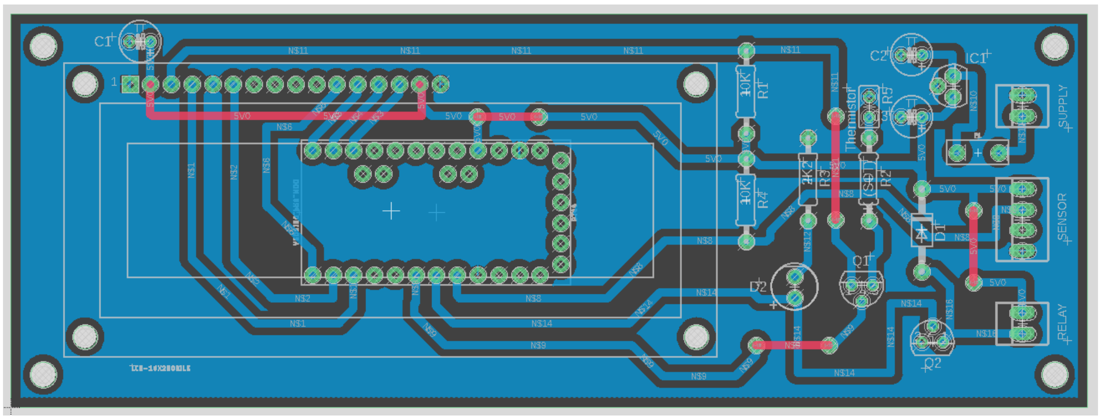
Dewpoint Monitor and Heating Controller

The electronics are based on an Arduino Pro Mini module which is connected to a DHT22 moisture and humidity monitor. The Arduino reads the DHT22 data stream and from this calculates the dewpoint. With some simple maths the Arduino compares the calculated dewpoint and the ambient temperature and when these close to within a defined limit the Arduino turns on a relay to power an external heater. The Arduino code also flashes the display ON and OFF once the dewpoint is triggered.

The DHT22 also provides a heat intensity value which is like a windchill factor. All these parameters are displayed on the LCD display as shown below.



After breadboarding the design, I committed it to a printed circuit board using the Fusion 360 Electrical module. Fusion produces Gerber files which I transformed to GCode using FlatCAM. The GCode was used to mill the printed circuit board on my Tormach PCNC440 CNC milling machine. I also used Fusion 360 to give me the milling codes to mill all the cut-out holes in the enclosure and the internal mounting plate.



The circuit diagram together with the Arduino code is attached at the end of this write up. Here are a few additional explanatory notes : -

The Arduino comes in various versions for 5V and 3V operation and with 8MHz or 16MHz clock speeds. I used the 5V / 16MHz version as I had these to hand and 5V operation best suited the intensity of the LCD display. Note that you will need a CH340G USB to serial converter to load the code into the Arduino. There are no changes to the Arduino code for these variants but you will need to tell the Arduino IDE which version you are using before trying to upload code.

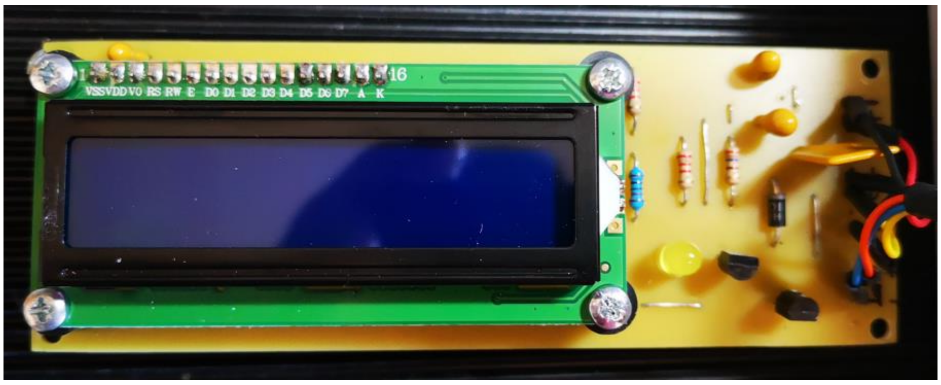
The PCB allows for 0.1" pitch Molex 254 series connectors but I opted for just terminal pins for the connections.

The DHT22 is a more accurate version of the DHT11 sensor. Both are equally useful for this application but I opted for the DHT22 version. Note that you must modify the Arduino code depending on which one you use. I chose to have the sensor connected remotely via a cable. Note that the DHT22 is also more tolerant of longer cable lengths. I made a simple protecting 3D printed enclosure to mount the sensor in. Alternately the sensor could be directly mounted on the box.

The PCB provides a switched output to drive a 5V relay. I opted to use a solid-state relay rather than a mechanical relay but either can be used providing attention is given to the rating. Note there is a back emf protection diode on the PCB should a mechanical relay be used.

The PCB has a small 78L05 regulator fitted to the board providing 5V to the electronics from any available higher voltage source. I had a spare 5V power supply to hand, which at 1A was totally overrated for this application. I replaced the onboard regulator with a link and fed the 5V directly to the electronics. The board consumes under 50mA excluding the relay current.

The PCB shown has an inline electronic fuse fitted and the AC inlet to the box is fused.



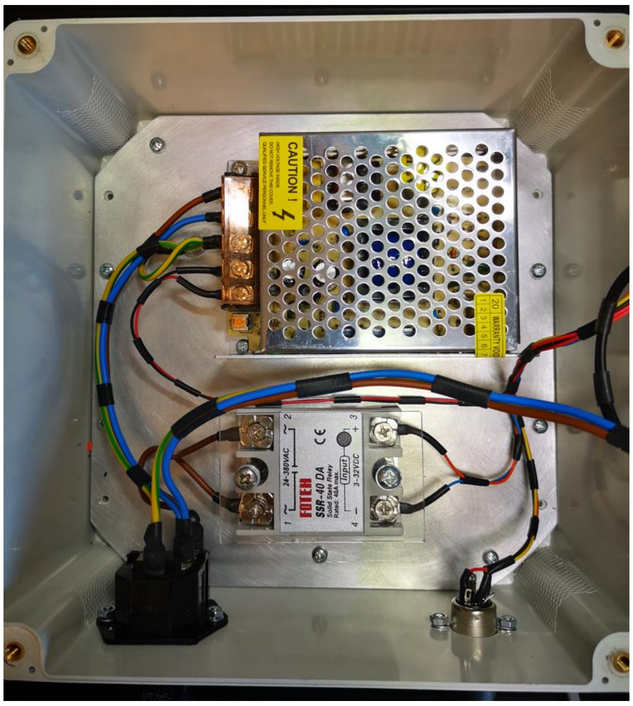
Not wanting to have additional holes in the box front panel, I used Araldite to glue four countersink screws to the inside face of the front panel and mounted the display off these.

The LCD is a readily available 16x2 character display. There are bezels available for this style but I opted to 3D print a bespoke bezel. I made four 11mm spacers to mount the LCD screen onto the PCB.

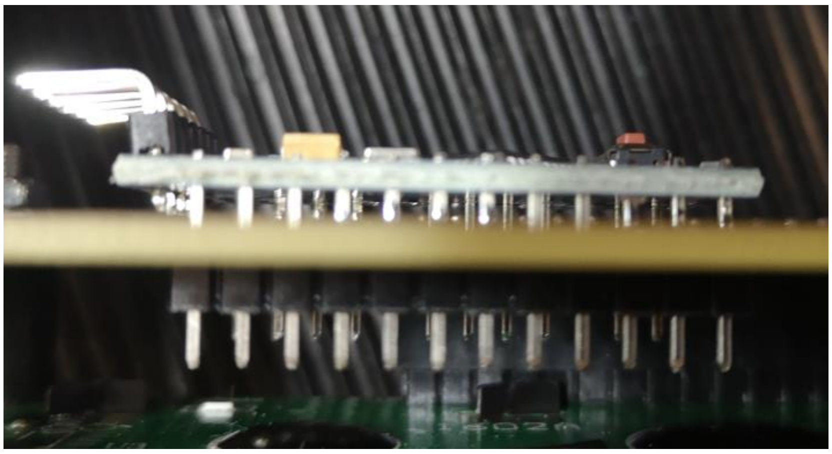
The display and the relay switching are both switched electronically using FET devices. I used 2N7000 TO92 packaged devices but any switching FET will suffice or indeed any digital switching transistor such as the DTC144. Depending on which device is chosen for this, the display brightness will change due to the ON resistance of the device. This is adjusted by selecting the SOT value in the source/emitter of the driver device. Using the 2N7000 this worked out at 270 ohms. I also allowed for a thermistor in case the display brightness changed with temperature but I have not used it and linked it out.

The box was a standard EBay offering and did not come with a mounting plate. Using Fusion 360 I made various paper templates until I got the shape and mounting holes correctly dimensioned. I then machined the plate in aluminium.

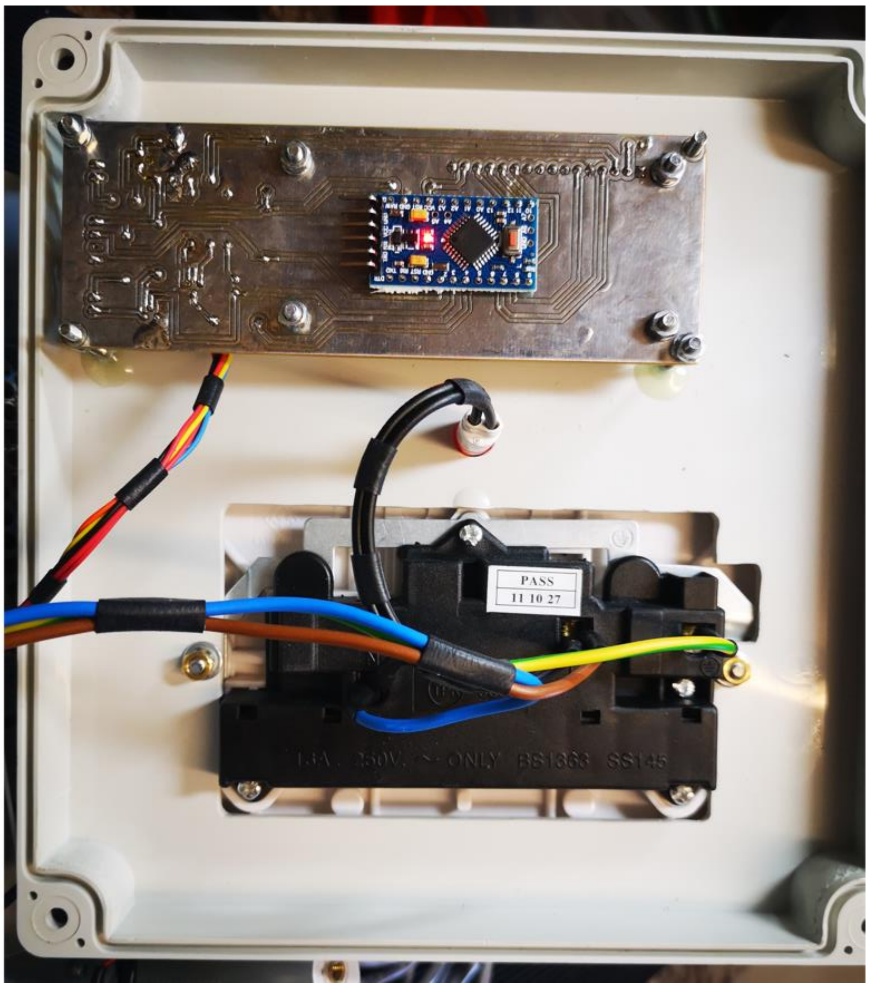
The solid-state relay has a metal backing plate that acts as a heatsink in high current switching applications. I used heatsink compound (aka "toothpaste") to make good thermal contact to the mounting plate but even switching a 3kW heater I could not detect any heating rise in the relay or the mounting plate. I made some simple hexagon spacers to mount the relay and these allowed me to fit a Perspex protection plate over the contact terminals.



I engineered the PCB layout to be a single sided board with links. This means only the trackside traces need to be milled. It does however complicate mounting the Arduino and soldering it in place. To overcome this, I mounted and soldered the single in line connector strips first and I then mounted the Arduino on these but slightly spaced off the PCB surface before soldering the pins into the Arduino board. If the board had been designed as a double-sided board this could have been simplified.



NOTE that if there is demand I could get professionally made double sided boards produced.



As can be seen, I like using Hellerman sleeves in my cable forms. This is far from an ideal production method as it is slow and tedious. It does allow you to be flexible in the cable routing and does allow mistakes to be easily rectified. You do need Hellerman pliers for this technique and I use DC4 silicon grease as the lubricant.

Conclusion

The project is quite straightforward (once you get the software correct ..) and offers a simple way to protect the workshop under damp cold conditions. The trip point margin can be changed by altering one parameter in the Arduino code.

Useful Related Links on Woody's Workshop

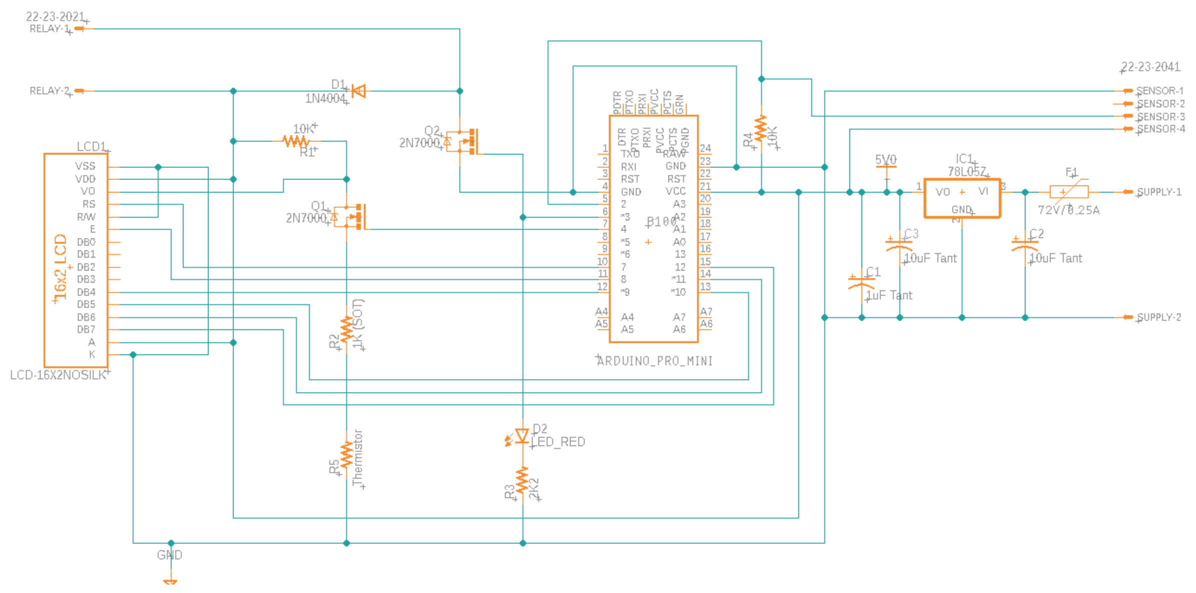
https://altrish.co.uk/2019/10/08/installing-an-externally-mounted-diesel-heater-for-heating-the-workshop/

https://www.wirelesstag.net/index.html

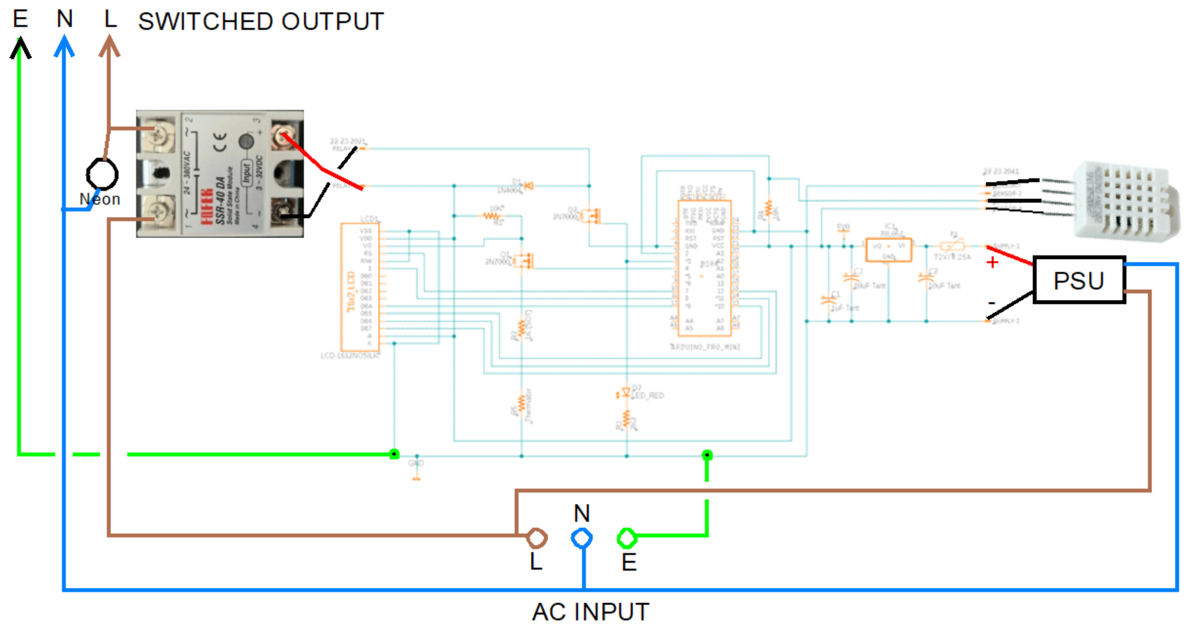
https://altrish.co.uk/2018/08/28/wireless-tag-technology-for-remote-sensing-and-security/

https://altrish.co.uk/2021/08/31/external-battery-pack-to-extend-the-battery-life-on-wireless-tags/

Circuit



External Connections



Arduino Code

```
// Temperature, Humidity and Dewpoint Display Module with alarm trip
// Created for Andover Model Engineering Society workshop protection system
// Author - Alan Wood, Woody's Workshop
// www.altrish.co.uk
// Version 1 - November 2021


// Libraries Setup

#include <LiquidCrystal.h>                  // LCD driver
#include <math.h>                           // Maths library
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);     // (rs,e,d4,d5,d6,d7) to be LCD interface connections
#include <DHT.h>                            // DHT11/DHT22 library

// Definitions

#define DHTPIN 2                            // Arduino pin D2 to be sensor input
DHT dht(DHTPIN, DHT22);                     // Change to DHT 22 or DHT11 sensor
#define DHTTYPE DHT22                       // Change to DHT 22 or DHT11 sensor

// Pin Assignments

int alarmLed = 3;                           //  Alarm output pin
int display_enable = 4;                     //  Display brightness enable (0V is ON)

// User Set Values

int trip_level = 5;                         //  Set trigger margin value between Temp and Dewpoint
unsigned long previousMillis = 0;           //  Used to set on/off flashing of display on dewpoint trip
const long interval = 2000;                 //  Flash rate for display once dewpoint triggered

void setup()
 {

   // Initialise

 Serial.begin(9600);                        //  Set serial comms baud rate
 dht.begin();                               //  Initialise sensor
 lcd.begin(16, 2);                          //  Initialise LCD

delay(2000);

   // Display and Serial Initialise

 Serial.println("ADMES Dewpoint Monitor");

   // Alarm Output, Display Control, Opening Message

 pinMode(alarmLed, OUTPUT);                 //  Assigns alarm output
 digitalWrite(alarmLed, LOW);               //  Alarm output start condition
 pinMode(display_enable,OUTPUT);            //  Assigns display enable output
 digitalWrite(display_enable,HIGH);         //  Display enable start condition (drives FET ON)

 lcd.setCursor(0, 0); lcd.print("      WORKSHOP      ");
 lcd.setCursor(0, 1); lcd.print("DEWPOINT MONITOR");
 delay (3000);
 }
```

```
void loop()
 {
   // Decode values from sensor serial stream

 float H = dht.readHumidity();                    //  Humidity (%)
 float C = dht.readTemperature();                 //  Temperature (C)
 float F = dht.readTemperature(true);             //  Temperature (F)


   // DHT Sensor Error Check

 if (isnan(H) || isnan(C) || isnan(F))
   {
  Serial.println("Reading failed from DHT sensor!");
  lcd.setCursor(0, 0); lcd.print("                ");
  lcd.setCursor(0, 0); lcd.print(" Read fail ");
  lcd.setCursor(0, 1); lcd.print("                ");
  lcd.setCursor(0, 1); lcd.print("Sensor Fail!");
  return;
   }

  //  Calculations based on Sonnatag Dewpoint values with sub calculation steps done as a cross check

 float number = H/100;                                  // Step by step calc of humidity/100
 float LN = log(number);                                // Natural Log of value of above calc
 float Sonntag = LN +((17.62*C)/(243.12+C));            // Sonntag value
 float DewPoint = (243.12*Sonntag)/(17.62-Sonntag);     // Dewpoint calc from Sonntag value
 float HiF = dht.computeHeatIndex(C, H);                // Heat Index from sensor stream (Celsius)


   //  Dewpoint alarm detection

 if ((C - trip_level) <= DewPoint)                      //  Check to see if Dewpoint is at trip point
   {digitalWrite(alarmLed, HIGH);}                      //  True sets output high
  else                                                  //  or if False
   {digitalWrite(alarmLed, LOW);                        //  Keeps alarm output low
   digitalWrite(display_enable,HIGH);}                  //  Ensures display stays on after trip

   //  Serial monitor

 Serial.print("Data Valid : ");                         //  prints sample OK
 Serial.print(C,1);       Serial.print(" *C, ");        //  prints Celsius temperature
 Serial.print(H,1);       Serial.print(" %, ");         //  prints humidity percentage
 Serial.print((int)HiF);  Serial.print(" Heat index, ");  //  prints heat index temperature
 Serial.print (LN,2);     Serial.print(" LogN, ");      //  prints LOGn value as cross check to 2dp
 Serial.print(Sonntag,2); Serial.print("  Sonn  ");     //  prints Sonntag sub equation value to 2dp
 Serial.print(DewPoint,1); Serial.println(" DewP, ");   //  prints dewpoint % from calc to 1dp

   //  LCD Address Matrix

 lcd.setCursor(0,  0); lcd.print("TMP");
 lcd.setCursor(4,  0); lcd.print(C);
 lcd.setCursor(6,  0); lcd.print((char)223); lcd.print("C");     //  this is degree symbol
 lcd.setCursor(8,  0); lcd.print(" DP ");
 lcd.setCursor(12, 0); lcd.print(DewPoint);
 lcd.setCursor(14, 0); lcd.print((char)223); lcd.print("C");     //  this is degree symbol
 lcd.setCursor(0,  1); lcd.print("HUM ");
 lcd.setCursor(4,  1); lcd.print(H);
```

```
  lcd.setCursor(6,  1); lcd.print("%  ");
  lcd.setCursor(8,  1); lcd.print(" HI ");
  lcd.setCursor(12, 1); lcd.print(HiF);
  lcd.setCursor(14, 1); lcd.print((char)223); lcd.print("C");        //  this is degree symbol

  // Sub routine to flash display ON and OFF if Dewpoint has been triggered

unsigned long currentMillis = millis();                   // checks clock time

if (digitalRead(alarmLed) == HIGH)                        // Checks to see if alarm is triggered

  if (currentMillis - previousMillis >= interval)         //  Checks to see if clock period >value
  {
  previousMillis = currentMillis;                         // resets count

  if (digitalRead(display_enable) == HIGH)                //  Toggles display ON & OFF
   {digitalWrite (display_enable,LOW);}
   else {digitalWrite (display_enable,HIGH);}
  }

delay(2000);                                              // this delay is necessary to allow
                                                          // sampling time for DHT sensors

  }
```

CODE ENDS

<u>Dewpoint Formula</u>

The dew point is calculated according to the following formula:

$$Ts = (b\alpha(T,RH)) / (a - \alpha(T,RH))$$

where:

Ts is the dew point;
T is the temperature;
RH is the relative humidity of the air;
a and b are coefficients.

The Sonntag90 constant values are : - -

$a = 17.62$ and $b = 243.12°C$;

and this is the final formula needed to define α : -

$$\alpha(T,RH) = \ln(RH/100) + aT/(b+T).$$