```
/*******************************
  Parts of the code are based off the adafruit_AHTx0 library example and utilizes functions from the wire library, arduino library, and online sources listed below.
  -millis implmentation: https://www.baldengineer.com/arduino-how-do-you-reset-millis.html
  -serialEvent implementation https://www.arduino.cc/en/Tutorial/BuiltInExamples/SerialEvent
  -Arduino and raspberry pi communication: https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/
  -Muliplexer tcaselect function: https://www.bluedot.space/tutorials/connect-multiple-sensors-using-i2c-multiplexer/
*******************************/
#include <Adafruit_AHTX0.h>
#include <Wire.h>
#define TCAADDR 0x70
#define weightC (float)0.3 //Weight constant for exponential filter
#define boxDelay (unsigned long)5000 //In ms. Used to set delay time for after the box is closed. This is to wait for temperature to stablize inside container
#define mesDelay (unsigned long)10000 //*In ms. This is for how long an error message should wait before senidng again.

//Sets up the sensor objects
Adafruit_AHTX0 sensor;
sensors_event_t humd, temp;

//Boolean variables to hold sensor state (ON/OFF)
bool sensor1State;
bool sensor2State;
//Boolean variable to hold box state(Open/Closed). Starts off as open to prevent the code from running at the beginning before getting a signal from pi
bool boxOpen=true;

//Temperature and humidity variable for each sensor
```

```
float t1,h1,t2,h2,wt=0,wh=0;//Current temp/humd reading and weighted temp/humd variables

float tavg,havg; //Averaged temp/humd varaibles

float idealTU=70,idealTL=20,idealHU=80,idealHL=20  ; //Variables for ideal temp/humd values


//Time variables used to keep track of time in order to prevent the arduino from spamming the raspberry pi with warnings on each loop

unsigned long countH=0, countT=0,countD=0,countC=0,countS=0;//countH: humidity, countT: temperature, countD=

/*NOTE: This timer is applied for each condition indvidually so if it detects an issue

with temperature at T=10s and humidity issue at T=20s, the temperature warning

message will send at T=25s (assuming 15s delay), and humidity at T=35s. */


//Variable to hold the input from pi

char message;

//END OF VARIABLE SETUP


/*

* tcaselect: Communicates with the tca9548a to select the correct

* i2c line for the sensor

*/

void tcaselect (uint8_t i) {

 if (i > 7) return;

 Wire.beginTransmission(TCAADDR);

 Wire.write(1 << i);

 Wire.endTransmission();

}

/*

* filterInput: Takes the averaged value temperature and humidity of the two sensors and

* uses the expoential filter to reduce noise in data
```

```
*/
float filterInput()
{
  wt = weightC * tavg + (1 - weightC) * (wt);
  wh = weightC * havg + (1 - weightC) * (wh);
}

/*
* checkSensors: Checks if the value readings on the sensors are valid
* If the sensors are reading significantly different values, it may indicate an issue
* with the sensor or the box
*/
void checkSensors() {
  if (abs(t1 - t2) > 15 || (abs(h1 - h2) > 30))
  {
    if ((unsigned long)(millis() - countS) > 20000) {
      Serial.println(F("Sensor reading abnormal"));
      countS = millis();
    }
  }
}

/*
* CheckConnect: Checks for connection status and populates the humd and temp with data from the sensor if connected.
* The getEvent provided by the library returns true if connected and collects the data from sensor if it is, removing
* the need to call getEvent again in the void loop.
*/
```

```
bool checkConnect(){
  if(!(sensor.getEvent(&humd,&temp))){
    if((unsigned long)(millis()-countD)>mesDelay){
      Serial.println(F("Sensor  disconnected"));
      countD = millis();
    }
    return false;
  }
  else{
    return true;
  }
}
/*
* serialEvent: Special function that runs after each void loop which
* checks for any serial inputs it may have recieved and depending  on
* the message recieved, it will change certain variables.
* If c is recieved, it will indicate that the box has been closed. If
* o is recieved, the arduino will know that the box is open
*/
void serialEvent(){
  while(Serial.available()){
    message = Serial.read();
    switch(message){
    case 'c':
      countC=millis();
      boxOpen=false;
      break;
    case 'o':
      boxOpen=true;
```

```
      break;
      /* Additional characters may be added for changing settings in the code dynamically
       * such as temperature range, boxDelay period, messageDelay period, and etc.
    case 'H':
      idealTU = 70;//Not likely to reach this temp(from testing). If it reaches this temp or above, may incicate that something is wrong
      idealTL = 20;
      idealHU = 70;
      idealHL = 20;
      break;
      */
    default:
      Serial.println(F("Invalid input"));
      break;
    }
  }
}
void setup() {
  Serial.begin(19200);
  while(!Serial){
    delay(10);
  }
  //Initalizes the sensors
  tcaselect(0);
  sensor1State = sensor.begin();
  tcaselect(1);
  sensor2State = sensor.begin();
  //Checks if sensors are available
  if (!sensor1State) {
```

```
  Serial.println(F("Sensor  1 not detected"));
}
if(!sensor2State){
  Serial.println(F("Sensor  2 not detected"));
}
//If both sensors are available, it will wait for the 'c' input to start
if(sensor1State&&sensor2State){
  Serial.println(F("Setup  successful"));
  while(boxOpen==true){
  serialEvent();
  delay(300);
  }
  //Presamples data to prevent the sensor from sending unideal condition warnings
  for (int i=0;i<25;i++){
    tcaselect(0);
    sensor1State = checkConnect();
    if(sensor1State){
    t1 = temp.temperature;
    h1 = humd.relative_humidity;
    }
    tcaselect(1);
    sensor2State = checkConnect();
    if(sensor2State){
    t2 = temp.temperature;
    h2 = humd.relative_humidity;
    }
    if(sensor1State&&sensor2State){
    tavg = (t1+t2)/2;
    havg = (h1+h2)/2;
```

```
    filterInput();
    }
   }
  }
}

void loop() {
 //Stops the program from running while the box is open
 while(boxOpen==true){
  serialEvent();
  delay(300);
 }
 //Delays the program from running when the box has just recently been closed to wait
 //until the conditions inside the box is stabilized
 while((unsigned long)(millis()-countC<boxDelay)){
  delay(300);
 }
 //Checks if sensors are connected
 tcaselect(0);
 sensor1State = checkConnect();
 //If connected, it will take the data from sensors
 if(sensor1State){
  t1 = temp.temperature;
  h1 = humd.relative_humidity;
 }
 tcaselect(1);
 sensor2State = checkConnect();
 if(sensor2State){
  t2 = temp.temperature;
```

```
  h2 = humd.relative_humidity;
 }

 //if both sensors are detected, it begins to process that data, averaging out the sensor
 //reading and using the filterInput function.
 //It also checks if the readings are abnormal, but will not prevent the program from
 //running if is abnormal. It will provide a warning message.
 if(sensor1State&&sensor2State){
  checkSensors();
  tavg = (t1+t2)/2;
  havg = (h1+h2)/2;
  filterInput();
  //Checks if the filtered data is outside of the presest ideal conditions and
  //returns warning message if it is
  if(wt>idealTU||wt<idealTL){
   if((unsigned long)(millis()-countT)>mesDelay){
    Serial.println(F("Temperature  not ideal"));
    countT = millis();
    }
   }
   if(wh>idealHU||wh<idealHL){
    if((unsigned long)(millis()-countH)>mesDelay){
     Serial.println(F("Humidity  not ideal"));
     countH = millis();
    }
   }
  }
 delay(20);
}
```