Temperature and Humidity Sensor on an Arduino

Though this tutorial is based exclusively on the usage of Temperature and humidity sensors or just a temperature sensor such as the KY-028, other tutorials may make usage of these temperature sensors associated with other sensors that may be the focus of those tutorials

A. DHT11

Introduction

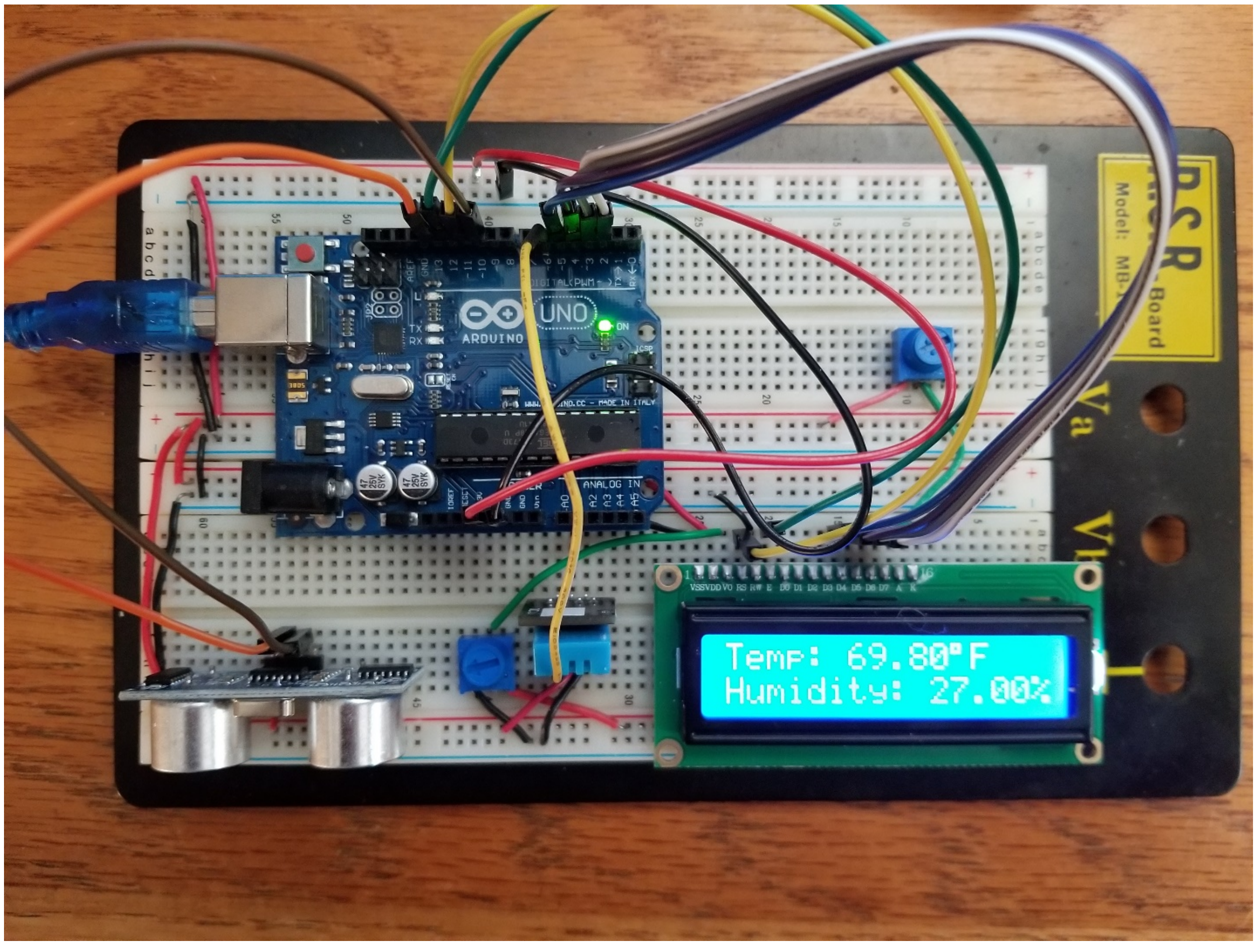


Fig 1. Test circuit to measure and display temperature and humidity

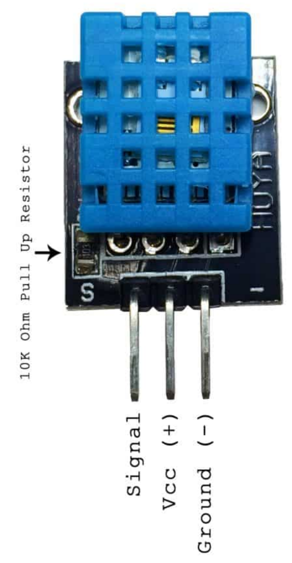
The DHT11 and DHT22 humidity and temperature sensors have been manufactured to be user friendly when connected to an Arduino or other microcontrollers. They are perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

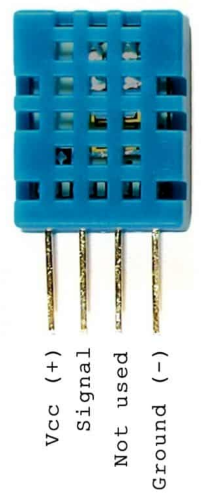
DHT11 vs DHT22

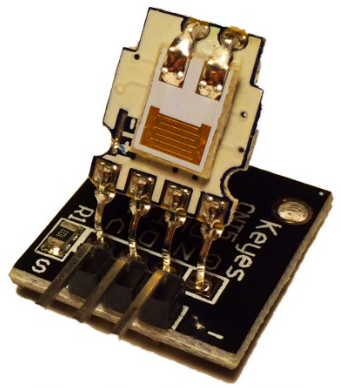
There are two versions of the DHT sensor, which look a bit similar and have the same pinout, but have different characteristics. Here are the specifications:

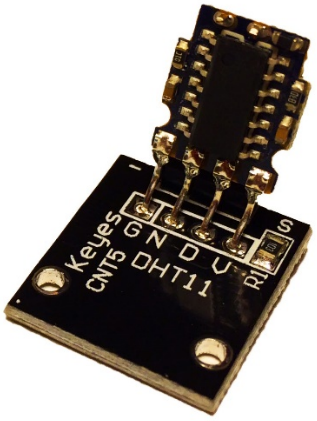
DHT11

- Ultra low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing






Front view with cover removed

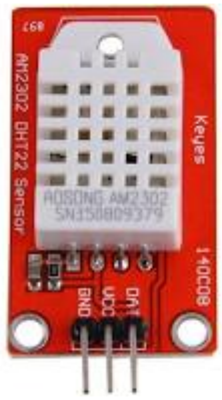

Rear view with cover removed

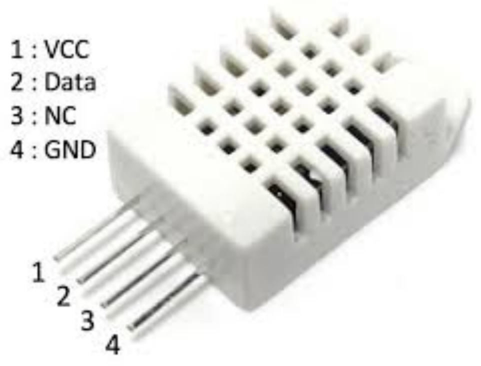
You have to be careful with the information about the pin labels. For example, the sensor shown in Fig 1 comes with the signal pin on the left side (as seen on Fig 1), followed by Vcc, then ground. I have another sensor well labeled since it does not follow this protocol, where the left pin ic Vcc, the middle pin is the signal pin, and the right pin is ground.

In the wired version (PCB), the board includes a surface mounted 10K Ohm pull up resistor for the signal line (between the signal line and the V_{CC}) to keep the signal level high by default.

DHT22 / AM2302 (Wired version)

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- Body size 15.1mm x 25mm x 7.7mm
- 4 pins with 0.1" spacing





It is possible that the wired version of the DHT22 follows the protocol of assigning the middle pin to the signal line (however, in the case of this 3-pin version, V_{CC} is the middle pin, ground is the left pin, and data is the right pin). Make sure you verify the pinout of your sensor.

As seen from the specifications, the DHT22 / AM2302 is more accurate and better over a larger range of temperatures, and the complete range of relative humidity. Both use a single digital pin and are slow in that you can't query them more than once every second or two; in fact, the more performing DHT22 is slower as measurements can only be obtained every 2 seconds, rather than 1 second for the DHT11.

Relative Humidity?

There are 3 definitions of humidity: absolute humidity, relative humidity, and specific humidity. We will be interested in relative humidity since this is what the DHT11 measures.

Wikipedia states that The relative humidity (RH or Φ) of an air-water mixture is defined as the ratio of the partial pressure of water vapor p_{H_2O} in the mixture to the equilibrium vapor pressure of water $p^*_{H_2O}$ over a flat surface of pure water at a given temperature.

$$\Phi = \frac{p_{H_2O}}{p^*_{H_2O}} * 100\%$$

In other words, relative humidity is the amount of water vapor in the air as a percentage of the saturation point of water vapor in air (maximum amount of water that can be held by the air), given the same temperature (temperature has a substantial effect on the humidity level). At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew. Condensation, then, occurs at 100% relative humidity.

The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

Measurement of Humidity and Temperature

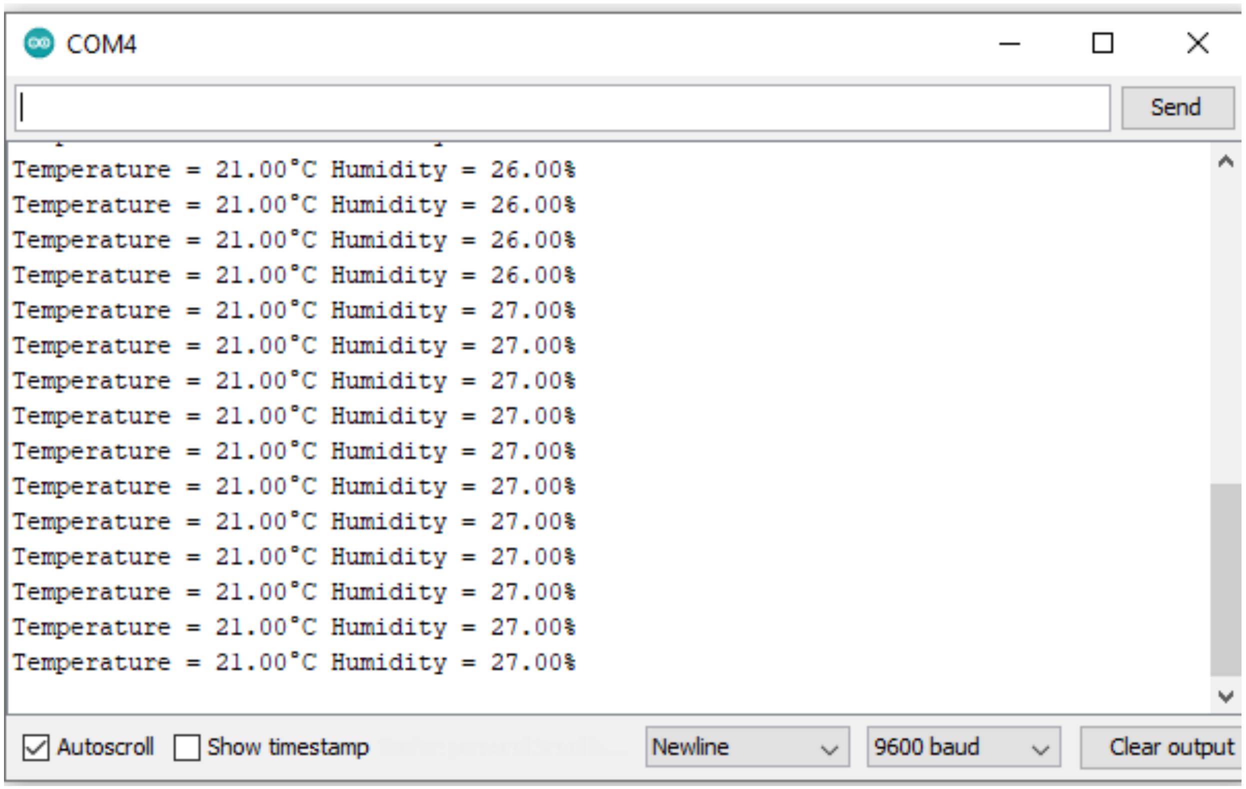
The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes (more conductive), while lower relative humidity increases the resistance between the electrodes. More information is available on the net.

DHT11 Setup on an Arduino

Wiring the DHT11 to the Arduino is really easy, but the connections are different depending on which type you have.

Display Humidity and Temperature on the Serial Monitor and the LCD

Assuming that you have installed the DHTLib and the LiquidCrystal libraries, the following program displays the temperature in ˚C on the serial monitor and the in ˚F on the LCD. In both cases, the humidity is given in %.



```
Temperature = 21.00°C Humidity = 26.00%
Temperature = 21.00°C Humidity = 26.00%
Temperature = 21.00°C Humidity = 26.00%
Temperature = 21.00°C Humidity = 26.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
Temperature = 21.00°C Humidity = 27.00%
```

Note how complex the steps are to display on the serial monitor data, strings, and special characters in one line. If anybody can find a simpler way, please suggest it as we need the code to be simpler and shorter.

After it's installed, upload this example program to the Arduino and open the serial monitor:

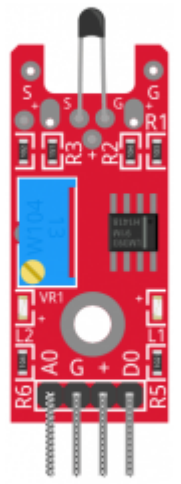
```
#include <dht.h>

#include <LiquidCrystal.h>
```

```
dht DHT;

LiquidCrystal lcd(11, 12, 5, 4, 3, 2);

#define DHT11_PIN 6


void setup(){

  Serial.begin(9600);

  lcd.begin(16, 2);

}


void loop(){

  int chk = DHT.read11(DHT11_PIN);


  Serial.print("Temperature = ");

  Serial.print(DHT.temperature);

  Serial.write(0xC2);

  Serial.write(0xB0);

  Serial.print("C ");

  Serial.print("Humidity = ");

  Serial.print(DHT.humidity);

  Serial.write(0x25);

  Serial.println("");


  lcd.setCursor(0,0);

  lcd.print("Temp: ");

  lcd.print(DHT.temperature*9/5+32);

  lcd.print((char)223);
  lcd.print("F");

  lcd.setCursor(0,1);

  lcd.print("Humidity: ");

  lcd.print(DHT.humidity);

  lcd.print("%");



  delay(2000);

}
```

The chapter that deals with the case of the Ultrasonic Range Finder on an Arduino will show how to use the readings from the DHT11 sensor to improve the accuracy of the evaluation of the distance between the sensors and an obstacle.

KY-028 Digital Temperature Sensor Module

The digital temperature sensor KY-028 for Arduino measures temperature variations through the variations of the resistance of a thermistor. A potentiometer is used to adjust the detection threshold on the digital interface.



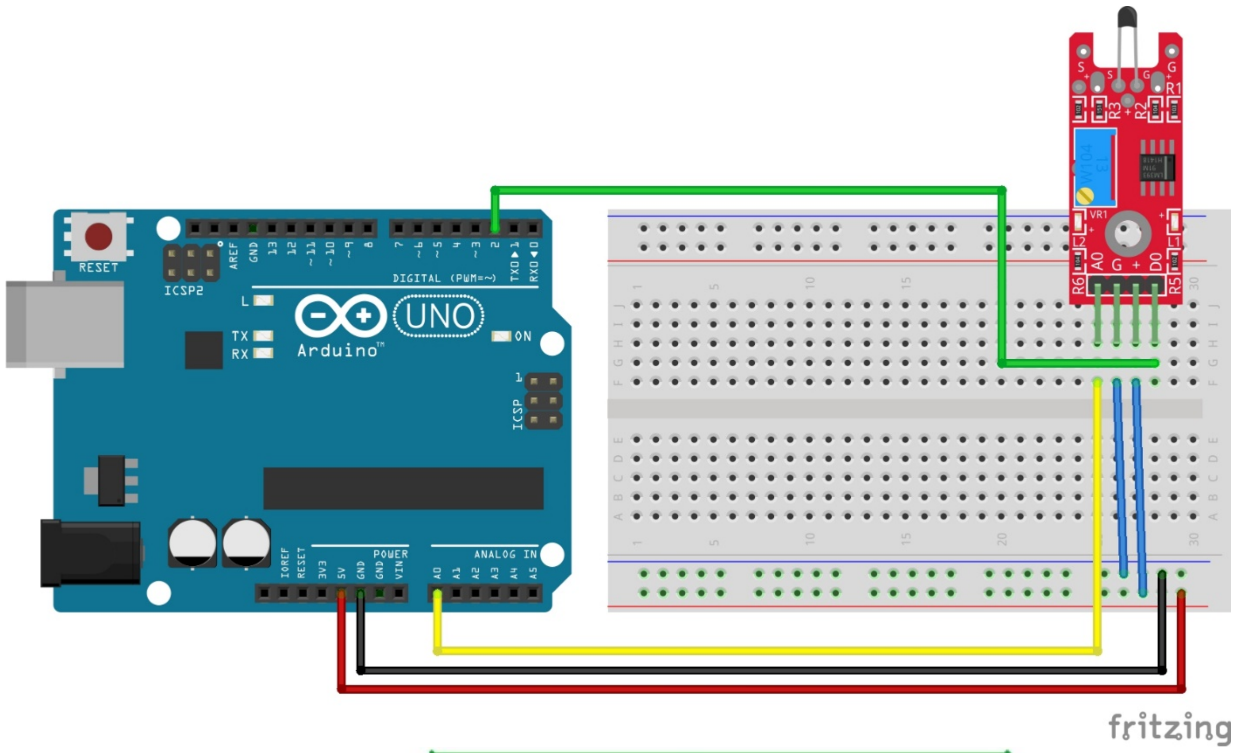
KY-028 Specifications

The KY-028 consists of a NTC (Negative Temperature Coefficient) thermistor, an LM393 dual differential comparator, a 3296W trimmer potentiometer, six resistors and two indicator LEDs. The board features an analog and a digital output.

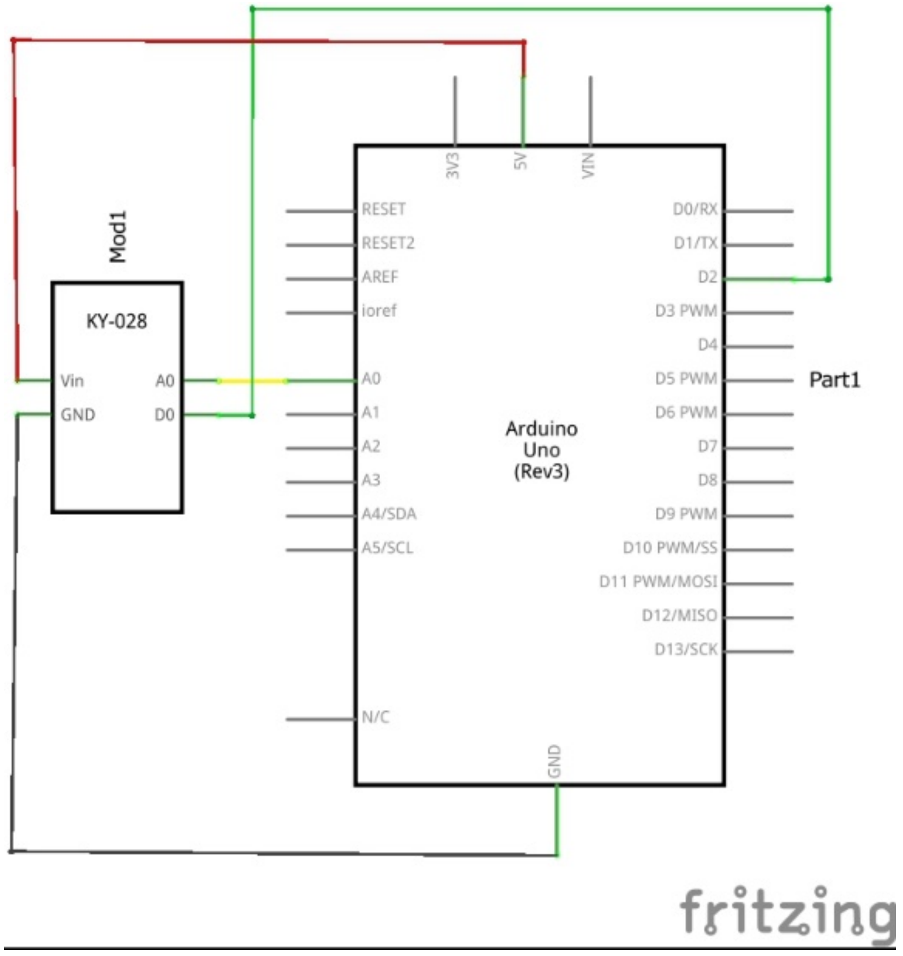
Operating Voltage	3.3V to 5.5V
Temperature measurement range	-55°C to 125°C [-67°F to 257°F]
Measurement Accuracy	±0.5°C
Board Dimensions	15mm x 36mm [0.6in x 1.4in]

KY-028 Connection Diagram

KY-028	Arduino
A0	A0
G	GND
+	5V
D0	2



RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
S
G
R1
R2
R3
VR1
A0
G
+
D0
R5
R6
fritzing



Mod1
KY-028
Vin
GND
A0
D0
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Arduino
Uno
(Rev3)
Part1
fritzing

KY-028 Arduino Code

When the temperature threshold is reached, the digital interface will send a HIGH signal turning on the LED on the Arduino (pin 13). Turn the potentiometer clock-wise to increase the detection threshold and counter-clockwise to decrease it.

The analog interface returns a numeric value that depends on the temperature and the potentiometer's position.

```
#include <ACI_10K_an.h>

int led = 13; // define the LED pin
int digitalPin = 2; // KY-028 digital interface
int analogPin = A0; // KY-028 analog interface
int digitalVal; // digital readings
int analogVal; //analog readings

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(digitalPin, INPUT);
  //pinMode(analogPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  // Read the digital interface
  digitalVal = digitalRead(digitalPin);
  if(digitalVal == HIGH) // if temperature threshold reached
  {
    digitalWrite(led, HIGH); // turn ON Arduino's LED
  }
  else
  {
    digitalWrite(led, LOW); // turn OFF Arduino's LED
  }

  // Read the analog interface
  analogVal = analogRead(analogPin);
  Serial.print("digitalVal = ");
  Serial.print(digitalVal); // print digital value to serial
  Serial.print("   AnalogVal = ");
  Serial.print(analogVal); // print analog value to serial

  Aci_10K an10k; //start an instance of the library
 //Aci_10K an10k(3.3,12);support for 3.3 volt board and/or 12bit analog read
resolution
 Serial.print("   temperature= ");
 Serial.println(an10k.getTemp(analogRead(0)));
 delay(1000);

}
```