DHT12 (i2c Cheap Humidity and Temperature Sensor), Fast Easy Usage

by xxreef

I like sensor that can be used with 2 wire (i2c protocol), but I love the inexpensive one.

This is an Arduino and esp8266 library for the DHT12 series of very low cost temperature/humidity sensors (less than 1$) that work with i2c or one wire connection.

Very usefully if you want use esp01 (if you use serial you have only 2 pin) to read humidity and temperature and display it on i2c LCD.

AI read that sometime seems that need calibration, but I have tree of this and get value very similar to DHT22. If you have calibration this problem, open issue on github and I add implementation.

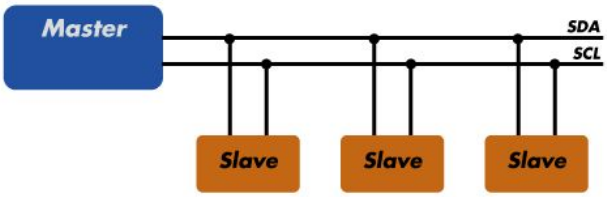
Step 1: How I2c Works

I2C works with it's two wires, the SDA(data line) and SCL(clock line).

Both these lines are open-drain, but are pulled-up with resistors.

Usually there is one master and one or multiple slaves on the line, although there can be multiple masters, but we'll talk about that later.

Both masters and slaves can transmit or receive data, therefore, a device can be in one of these four states: master transmit, master receive, slave transmit, slave receive.



Step 2: Library

You can find my library here.

To download.

Click the DOWNLOADS button in the top right corner, rename the uncompressed folder DHT12.

Check that the DHT folder contains DHT12.cpp and DHT12.h.

Place the DHT library folder your /libraries/ folder.

You may need to create the libraries subfolder if its your first library.

Restart the IDE.

Step 3: Behaivor

This libray try to emulate the behaivor of standard DHT library sensors (and copy a lot of code), and I add the code to manage i2c olso in the same manner.

The method is the same of DHT library sensor, with **some adding like dew point function**.

Step 4: I2c Usage

To use with i2c (default address and default SDA SCL pin) the constructor is:

```
DHT12 dht12;
```

and take the default value for SDA SCL pin.

(It's possible to redefine with specified contructor for esp8266, needed for ESP-01). or

```
DHT12 dht12(uint8_t addressOrPin)
```

```
addressOrPin -> address
```

to change address.

Step 5: One Wire Usage

To use one wire:

```
DHT12 dht12(uint8_t addressOrPin, true)
```

```
addressOrPin -> pin
```

boolean value is the selection of oneWire or i2c mode.

Step 6: Implicit Read

You can use It with "implicit", "simple read" or "fullread": Implicit, only the first read doing a true read of the sensor, the other read that become in 2secs. interval are the stored value of first read.

```
// The read of sensor have 2secs of elapsed time, unless you pass force parameter
  // Read temperature as Celsius (the default)
  float t12 = dht12.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f12 = dht12.readTemperature(true);
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h12 = dht12.readHumidity();


  // Compute heat index in Fahrenheit (the default)
  float hif12 = dht12.computeHeatIndex(f12, h12);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic12 = dht12.computeHeatIndex(t12, h12, false);
  // Compute dew point in Fahrenheit (the default)
  float dpf12 = dht12.dewPoint(f12, h12);
  // Compute dew point in Celsius (isFahreheit = false)
  float dpc12 = dht12.dewPoint(t12, h12, false);
```

Step 7: Simple Read

Simple read to get a status of read.

```
// The read of sensor have 2secs of elapsed time, unless you pass force parameter
bool chk = dht12.read(); // true read is ok, false read problem
```

```
// Read temperature as Celsius (the default)
float t12 = dht12.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f12 = dht12.readTemperature(true);
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h12 = dht12.readHumidity();

// Compute heat index in Fahrenheit (the default)
float hif12 = dht12.computeHeatIndex(f12, h12);
// Compute heat index in Celsius (isFahreheit = false)
float hic12 = dht12.computeHeatIndex(t12, h12, false);
// Compute dew point in Fahrenheit (the default)
float dpf12 = dht12.dewPoint(f12, h12);
// Compute dew point in Celsius (isFahreheit = false)
float dpc12 = dht12.dewPoint(t12, h12, false);
```

Step 8: Full Read

Full read to get a specified status.

```
// The read of sensor have 2secs of elapsed time, unless you pass force parameter
DHT12::ReadStatus chk = dht12.readStatus();
Serial.print(F("\nRead sensor: "));
switch (chk) {
case DHT12::OK:
 Serial.println(F("OK"));
 break;
case DHT12::ERROR_CHECKSUM:
 Serial.println(F("Checksum error"));
 break;
case DHT12::ERROR_TIMEOUT:
 Serial.println(F("Timeout error"));
 break;
case DHT12::ERROR_TIMEOUT_LOW:
 Serial.println(F("Timeout error on low signal, try put high pullup resistance"));
 break;
case DHT12::ERROR_TIMEOUT_HIGH:
 Serial.println(F("Timeout error on low signal, try put low pullup resistance"));
 break;
case DHT12::ERROR_CONNECT:
 Serial.println(F("Connect error"));
 break;
case DHT12::ERROR_ACK_L:
 Serial.println(F("AckL error"));
 break;
case DHT12::ERROR_ACK_H:
 Serial.println(F("AckH error"));
 break;
case DHT12::ERROR_UNKNOWN:
 Serial.println(F("Unknown error DETECTED"));
 break;
case DHT12::NONE:
 Serial.println(F("No result"));
 break;
default:
 Serial.println(F("Unknown error"));
 break;
}
```

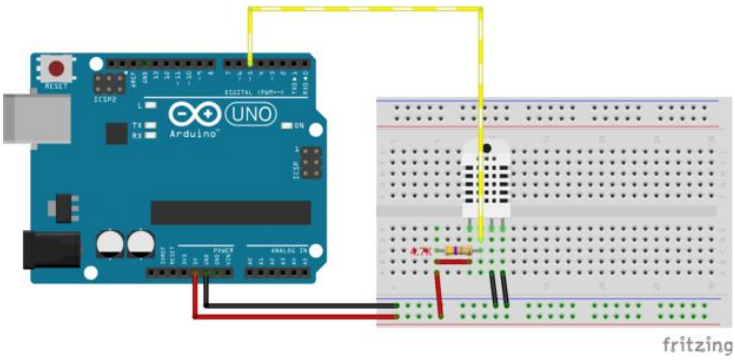
```
// Read temperature as Celsius (the default)
float t12 = dht12.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f12 = dht12.readTemperature(true);
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h12 = dht12.readHumidity();

// Compute heat index in Fahrenheit (the default)
float hif12 = dht12.computeHeatIndex(f12, h12);
// Compute heat index in Celsius (isFahreheit = false)
float hic12 = dht12.computeHeatIndex(t12, h12, false);
// Compute dew point in Fahrenheit (the default)
float dpf12 = dht12.dewPoint(f12, h12);
// Compute dew point in Celsius (isFahreheit = false)
float dpc12 = dht12.dewPoint(t12, h12, false);
```

Step 9: Connection Diagram

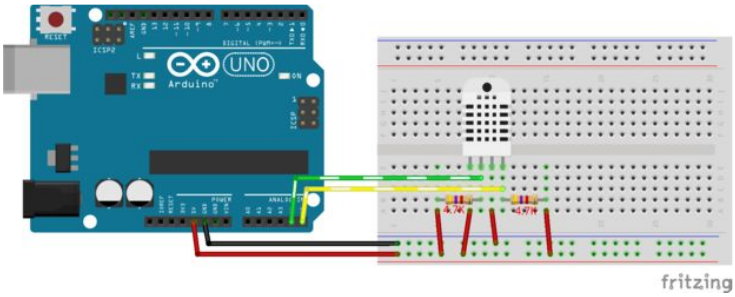
With examples, there are the connection diagram, it's important to use correct pullup resistor.

Thanks to Bobadas, dplasa and adafruit, to share the code in github (where I take some code and ideas).

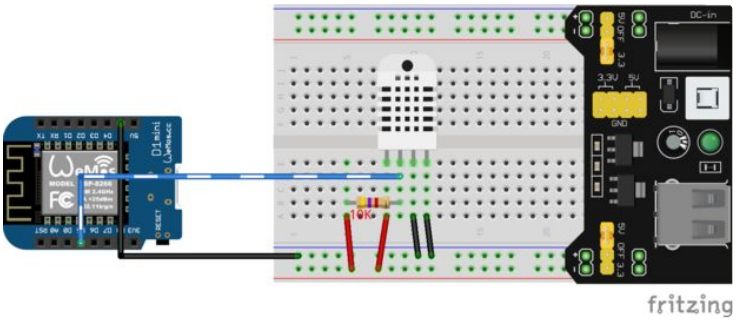
Step 10: Arduino: OneWire



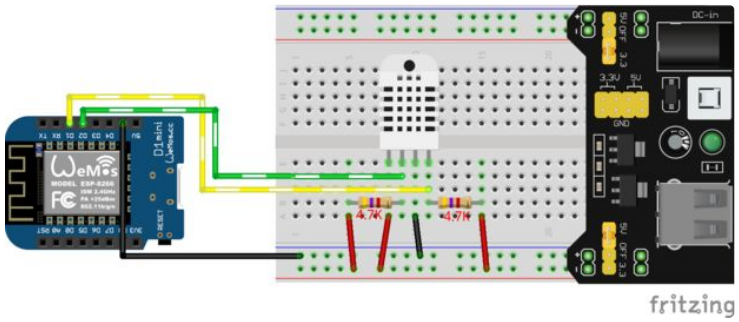
Step 11: Arduino: I2c



Step 12: Esp8266 (D1Mini) OneWire



Step 13: Esp8266 (D1Mini) I2c



Step 14: Thanks

i2c project series:

- Temperature humidity sensor
- Analog expander
- Digital expander