A DHT11 Class for Arduino.

Intro

The DHT11 is a relatively cheap sensor for measuring temperature and humidity. This article describes a small library for reading both from the sensor. The DHT22 is similar to the DHT11and has greater accuracy. However, this library is not suitable for the DHT21 or DHT22 as they have a different data format. Check DHTlib for support of these sensors.
This library is tested on a MEGA2560 and is confirmed working on an Arduino 2009.
Niesteszeck has made an interrupt-driven library for the DHT11 sensor.
Andy Dalton has made a modified version. Difference is that the DATAPIN is defined in the constructor, resulting in one dedicated object per sensor.

Connection

The DHT11 has three lines: GND, +5V and a single data line. By means of a handshake, the values are clocked out over the single digital line.
Datasheet: http://www.micro4you.com/files/sensor/DHT11.pdf

DHT11 library

The library proposed here is based upon earlier work of George Hadjikyriacou. SimKardcreated a new version, which I engineered together with him, resulting in the current 0.3.2 version. It is not backwards compatible with the earlier 0.2 version as temperature conversion and dewpoint calculations were removed from the class to keep the it as minimal as possible. This made the library quite a bit smaller. The sample sketch presented below includes the dewPoint functions so one can still use them.
The class interface supports only one function for reading the humidity and temperature (both members of the class). The read() function verifies the data transmission's checksum. Furthermore, it has a timeout function (which may be improved). The class is kept simple and, with one instance, it's possible to read multiple sensors, provided that each sensor has a separate pin.
The read() function returns
• DHTLIB_OK (0): The sensor samples and its checksum are OK.
•DHTLIB_ERROR_CHECKSUM (-1): The checksum test failed. This means that data was received but may not be correct.
•DHTLIB_ERROR_TIMEOUT (-2): A timeout occurred, and communication has failed.

DewPoint functions

The sample sketch shows two dewPoint functions. One more professional (NOAA-based) and a faster one called dewPointFast(). The latter is smaller and 5x faster than the NOAA one, with has a maximum error of 0.6544 C. As the DHT11 sensor has ~ 1% accuracy, the fast version might be accurate enough for most applications.

Usage

A sketch shows how the library can be used to read the sensor:

```
//
//    FILE:  dht11_test1.pde
// PURPOSE: DHT11 library test sketch for Arduino
//

//Celsius to Fahrenheit conversion
double Fahrenheit(double celsius)
{
```

```
        return 1.8 * celsius + 32;
}

// fast integer version with rounding
//int Celcius2Fahrenheit(int celcius)
//{
//  return (celsius * 18 + 5)/10 + 32;
//}


//Celsius to Kelvin conversion
double Kelvin(double celsius)
{
        return celsius + 273.15;
}

// dewPoint function NOAA
// reference (1) : http://wahiduddin.net/calc/density_algorithms.htm
// reference (2) : http://www.colorado.edu/geography/weather_station/Geog_site/about.htm
//
double dewPoint(double celsius, double humidity)
{
        // (1) Saturation Vapor Pressure = ESGG(T)
        double RATIO = 373.15 / (273.15 + celsius);
        double RHS = -7.90298 * (RATIO - 1);
        RHS += 5.02808 * log10(RATIO);
        RHS += -1.3816e-7 * (pow(10, (11.344 * (1 - 1/RATIO ))) - 1) ;
        RHS += 8.1328e-3 * (pow(10, (-3.49149 * (RATIO - 1))) - 1) ;
        RHS += log10(1013.246);

        // factor -3 is to adjust units - Vapor Pressure SVP * humidity
        double VP = pow(10, RHS - 3) * humidity;

        // (2) DEWPOINT = F(Vapor Pressure)
        double T = log(VP/0.61078);   // temp var
        return (241.88 * T) / (17.558 - T);
}

// delta max = 0.6544 wrt dewPoint()
// 6.9 x faster than dewPoint()
// reference: http://en.wikipedia.org/wiki/Dew_point
double dewPointFast(double celsius, double humidity)
{
        double a = 17.271;
        double b = 237.7;
        double temp = (a * celsius) / (b + celsius) + log(humidity*0.01);
        double Td = (b * temp) / (a - temp);
        return Td;
}


#include <dht11.h>

dht11 DHT11;

#define DHT11PIN 2

void setup()
{
  Serial.begin(115200);
  Serial.println("DHT11 TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
```

```
}

void loop()
{
  Serial.println("\n");

  int chk = DHT11.read(DHT11PIN);

  Serial.print("Read sensor: ");
  switch (chk)
  {
    case DHTLIB_OK:
                Serial.println("OK");
                break;
    case DHTLIB_ERROR_CHECKSUM:
                Serial.println("Checksum error");
                break;
    case DHTLIB_ERROR_TIMEOUT:
                Serial.println("Time out error");
                break;
    default:
                Serial.println("Unknown error");
                break;
  }

  Serial.print("Humidity (%): ");
  Serial.println((float)DHT11.humidity, 2);

  Serial.print("Temperature (°C): ");
  Serial.println((float)DHT11.temperature, 2);

  Serial.print("Temperature (°F): ");
  Serial.println(Fahrenheit(DHT11.temperature), 2);

  Serial.print("Temperature (°K): ");
  Serial.println(Kelvin(DHT11.temperature), 2);

  Serial.print("Dew Point (°C): ");
  Serial.println(dewPoint(DHT11.temperature, DHT11.humidity));

  Serial.print("Dew PointFast (°C): ");
  Serial.println(dewPointFast(DHT11.temperature, DHT11.humidity));

  delay(2000);
}
//
// END OF FILE
//
```

[Get Code]

In setup() The version string (a define) is displayed. This is for debugging purpose only.
In loop() the sensor is read and the fields temperature and humidity are filled. The return value of the read function is checked and displayed. Then the temperature and humidity is shown in various formats, and the dew point is calculated and displayed.

Notes

To use the library, make a folder in your SKETCHBOOKPATH\libaries with the name DHT11and put the .h and .cpp there. Optionally make a examples subdirectory to place the sample app. Be aware that the library will only be visible after restarting all instances of the Arduino IDE.