The playground is a publicly-editable wiki about Arduino.

Manuals and Curriculum
Installing Arduino on Linux
Board Setup and Configuration
Development Tools
Arduino on other Atmel Chips
Interfacing With Hardware
- Output
- Input
- User Interface
- Storage
- Communication
- Power supplies
- General

Interfacing with Software
User Code Library
- Snippets and Sketches
- Libraries
- Tutorials

Suggestions & Bugs
Electronics Technique
Sources for Electronic Parts
Related Hardware and Initiatives
Arduino People/Groups & Sites
Exhibition
Project Ideas
Languages

PARTICIPATE
- Suggestions
- Formatting guidelines
- All recent changes
- PmWiki
- WikiSandBox training
- Basic Editing
- Cookbook (addons)
- Documentation index

edit SideBar

A DHT11 Class for Arduino.

remarks & comments

Intro

The DHT11 is a relative 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 DHT11 and has a greater accuracy. However the library is not suitable for the DHT22 as it has a different dataformat. The protocol is similar and therefor a separate DHT22 library is planned.

The library is tested on a MEGA2560 and confirmed working on a Arduino 2009.

Connection

The DHT11 has three lines, GND, +5V and a single dataline. 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. SimKard created 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 class as minimal as possible. This made the lib quite a bit smaller. The sample sketch presented below includes the dewPoint functions so one can still use them.

The interface of the class supports only one function for reading the humidity and temperature, which are members of the class. The read() function verifies the checksum of the datatransmission. Furthermore it has a timeout function (which may be improved). The class is kept simple and with one instance it is possible to read multiple sensors, provided each sensor has a separate pin.

The read() function returns

- DHTLIB_OK (0) : if the sensor sample and its checksum is OK.
- DHTLIB_ERROR_CHECKSUM (-1) : if the checksum test failed. This means that data was received but may be incorrect.
- DHTLIB_ERROR_TIMEOUT (-2) : if a timeout occured, communication failed.

DewPoint functions

The sample sketch shows two dewPoint functions. One more professional (NOAA - based) and a faster one called dewPointFast(). This latter is smaller and 5x faster and has a maximum error of 0.6544 C compared to the NOAA one. 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;
}

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

// dewPoint function NOAA
// reference: http://wahiduddin.net/calc/density_algorithms.htm
double dewPoint(double celsius, double humidity)
{
        double A0= 373.15/(273.15 + celsius);
        double SUM = -7.90298 * (A0-1);
        SUM += 5.02808 * log10(A0);
        SUM += -1.3816e-7 * (pow(10, (11.344*(1-1/A0)))-1) ;
        SUM += 8.1328e-3 * (pow(10,(-3.49149*(A0-1)))-1) ;
        SUM += log10(1013.246);
        double VP = pow(10, SUM-3) * humidity;
        double T = log(VP/0.61078);   // temp var
        return (241.88 * T) / (17.558-T);
}

// delta max = 0.6544 wrt dewPoint()
// 5x 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/100);
        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 (oC): ");
  Serial.println((float)DHT11.temperature, 2);

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

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

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

  Serial.print("Dew PointFast (oC): ");
  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 DHT11 and 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.

Todo

- Test for full range of temperatures and humiditiy
- Handle one second delay after power up. // millis() > 1000 in read ?
  // No, would only makes sense first second and after that it is overhead.
  (my choice)

Done

- Add timeout code
- Remove float
  - Split off the dewPoint function
  - DHT11 int version
- 2012-mar-17: made version 0.4.0 ==>

made the lib 1.0 compatible (Wprogram.h vs Arduino.h)

Enjoy tinkering,

rob.tillaart@removethisgmail.com

dht11.h

```
//
//    FILE: dht11.h
// VERSION: 0.4.1
// PURPOSE: DHT11 Temperature & Humidity Sensor library for Arduino
// LICENSE: GPL v3 (http://www.gnu.org/licenses/gpl.html)
//
// DATASHEET: http://www.micro4you.com/files/sensor/DHT11.pdf
//
//     URL: http://arduino.cc/playground/Main/DHT11Lib
//
// HISTORY:
// George Hadjikyriacou - Original version
// see dht.cpp file
//

#ifndef dht11_h
#define dht11_h

#if defined(ARDUINO) && (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif

#define DHT11LIB_VERSION "0.4.1"

#define DHTLIB_OK                               0
#define DHTLIB_ERROR_CHECKSUM   -1
#define DHTLIB_ERROR_TIMEOUT    -2

class dht11
{
public:
    int read(int pin);
        int humidity;
        int temperature;
};
#endif
//
// END OF FILE
//
```

[Get Code]

dht11.cpp

```
//
//    FILE: dht11.cpp
// VERSION: 0.4.1
// PURPOSE: DHT11 Temperature & Humidity Sensor library for Arduino
// LICENSE: GPL v3 (http://www.gnu.org/licenses/gpl.html)
//
// DATASHEET: http://www.micro4you.com/files/sensor/DHT11.pdf
//
// HISTORY:
// George Hadjikyriacou - Original version (??)
// Mod by SimKard - Version 0.2 (24/11/2010)
// Mod by Rob Tillaart - Version 0.3 (28/03/2011)
// + added comments
// + removed all non DHT11 specific code
// + added references
// Mod by Rob Tillaart - Version 0.4 (17/03/2012)
// + added 1.0 support
// Mod by Rob Tillaart - Version 0.4.1 (19/05/2012)
// + added error codes
//

#include "dht11.h"

// Return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht11::read(int pin)
{
        // BUFFER TO RECEIVE
        uint8_t bits[5];
        uint8_t cnt = 7;
        uint8_t idx = 0;

        // EMPTY BUFFER
        for (int i=0; i< 5; i++) bits[i] = 0;

        // REQUEST SAMPLE
        pinMode(pin, OUTPUT);
        digitalWrite(pin, LOW);
        delay(18);
        digitalWrite(pin, HIGH);
        delayMicroseconds(40);
        pinMode(pin, INPUT);

        // ACKNOWLEDGE or TIMEOUT
        unsigned int loopCnt = 10000;
        while(digitalRead(pin) == LOW)
                if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;
```

```
        loopCnt = 10000;
        while(digitalRead(pin) == HIGH)
                if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

        // READ OUTPUT - 40 BITS => 5 BYTES or TIMEOUT
        for (int i=0; i<40; i++)
        {
                loopCnt = 10000;
                while(digitalRead(pin) == LOW)
                        if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

                unsigned long t = micros();

                loopCnt = 10000;
                while(digitalRead(pin) == HIGH)
                        if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

                if ((micros() - t) > 40) bits[idx] |= (1 << cnt);
                if (cnt == 0)   // next byte?
                {
                        cnt = 7;    // restart at MSB
                        idx++;      // next byte!
                }
                else cnt--;
        }

        // WRITE TO RIGHT VARS
        // as bits[1] and bits[3] are allways zero they are omitted in formulas.
        humidity    = bits[0];
        temperature = bits[2];

        uint8_t sum = bits[0] + bits[2];

        if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;
        return DHTLIB_OK;
}
//
// END OF FILE
//
```

[Get Code]