HTU21D Humidity Sensor Hookup Guide

CONTRIBUTORS: NATE

HTU21D Overview

The HTU21D is a low-cost, easy to use, highly accurate, digital humidity sensor. All you need is two lines for I²C communication and you'll have relative humidity readings such as "45.2%" or "23.1%" and very accurate temperature readings as a bonus!

Things you should know about this sensor:

- Uses the I²C interface
- Typical humidity accuracy of ±2%
- Typical temperature accuracy of ±0.3C
- Operates from 0 to 100% humidity but this sensor isn't recommended for harsh environments where it could come in contact with water (such as rain).
- 3.3V sensor - use inline logic level converters or 10k resistors to limit 5V signals
- Here's the datasheet

- Only one HTU21D sensor can reside on the I²C bus at a time

This sensor is ideal for environmental sensing and data logging. Perfect for a weather station or humidor control system. It is a very good replacement for digital humidity sensors such as the SHT15, SHT21, SHT25, HIH-4030, HIH6130 and capacitive humidity sensors such as the HH10D.

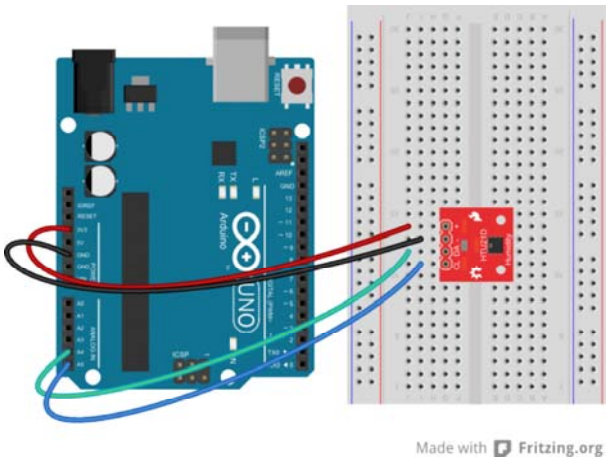
Suggested Reading

Things you might need to know:

- I²C Protocol
- Using the PCA9306 voltage translator
- Logic Levels
- Installing an Arduino library
- What are pull-up resistors?
- How to use a breadbaord

Hooking It Up

Wiring up the HTU21D humidity sensor is very easy! We recommend soldering four male headers to the breakout board and using the HTU21D in a breadboard.



Connections: Breakout board to Arduino

- VCC → 3.3V
- GND → GND
- SDA → A4
- SCL → A5

There are only four pins that need to be hooked up in order to start using this sensor in a project. One for VCC, one for GND, and two data lines for I²C communication. On an Arduino board connect the SDA pin on the

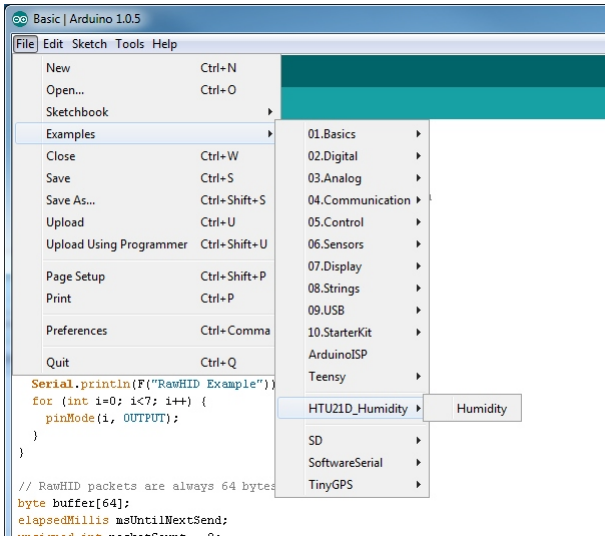
breakout board to A4 and SCL to A5. If you have a newer Arduino, you can connect the SDA and SCL lines directly to the SDA and SCL lines broken out on the Arduino headers.

This board runs at 3.3V. Be sure to power the board from the 3.3V pin! Because I²C is an open drain signal, there's no need to worry about level shifting the signal; the 3.3V signal will be adequate to communicate with the Arduino and the signal will never reach a dangerous level for the pins on the HTU21D.

Note: This breakout board has built in 4.7k pull up resistors for I²C communications. If you're hooking up multiple I²C devices on the same bus, you may want to disable these resistors.

HTU21D Library and Functions

We've written an Arduino library and some example code to make using the HTU21D easy to get up and running. Grab the HTU21D library for Arduino here. Copy and paste the folder "HTU21D_Humidity" into your Arduino libraries folder. Not sure how to install an Arduino library? Read here for more information.



The examples menu expanded to show Humidity example

Once the library is installed, open Arduino, and expand the examples menu. You should see the HTU21D_Humidity submenu.

Load this example onto the Arduino. Open the serial terminal at 9600bps. You will see the current humidity and temperature in the room!

Explanation of Functions:

The library and example code demonstrate some of the smaller functions supported by the HTU21D. These are generally not used by regular users but are documented here in case you need them:

- `myHumidity.readHumidity()` will return a float containing the humidity. Ex: 54.7
- `myHumidity.readTemperature()` will return a float containing the temperature in Celsius. Ex: 24.1
- `myHumidity.setResolution(byte: 0b.76543210)` sets the resolution of the readings.
- `myHumidity.check_crc(message, check_value)` verifies the 8-bit CRC generated by the sensor.
- `myHumidity.read_user_register()` returns the user register. Used to set resolution.

When you call the ReadHumidity or ReadTemperature functions you will get a float with the sensor reading or an error code:

- **57.8** is an example of a valid reading. 0.0 to 100.0 for humidity and -40.0 to 125.0 for temperature.
- **998** indicates that I2C timed out (100ms max). Check your connections.
- **999** CRC was wrong. The HTU21D calculates an internal CRC and transmits it along with the temperature and humidity readings. It's highly unlikely that you will *ever* see a bad CRC, but the library supports CRC checking.

`SetResolution()` allows the user to change the humidity and temperature resolution. The vast majority of users do not need to change the resolution. By default the sensor will be in its highest resolution settings. This function is useful if you need to decrease the amount of time between readings or to save power. See page 12 of the datasheet. As an example, to change the resolution to 11 bit RH and 11 bit temperature, you would call `myHumidity.SetResolution(0b10000001);` to set bit 7 and bit 0.

`check_crc()` inputs the message and check_value from the HTU21D and then does a 8-bit polynomial CRC to verify that the message the sensor sent was valid. It's extremely rare that you will see a bad CRC but if you're looking into how to decipher a 8-bit CRC, this is a good example. The HTU21D uses the same CRC as 1-wire products.

Disabling I2C Resistors

There are two 4.7k resistors on the breakout board that pull-up the SDA and SCL lines. If you've got other devices on the I^2C bus you may need to disable the on-board resistors. To do this you need to clear the solder jumper on the board. This will disconnect the resistors from VCC and from the I^2C bus.

Resources and Going Further

You should now have a good idea of how to add humidity and temperature sensing into your next project. Need some inspiration? Check out these other tutorials:

- Make an automated terrarium that manages heat and humidity levels.
- Build a humidor control box that maintains a constant humidity in a controlled space.

Resources:

- Example Arduino Sketch demonstrating how to talk to the HTU21D Humidity Sensor
- HTU21D Datasheet
- HTU21D Breakout Board Schematic
- HTU21D Breakout Board Eagle Files
- Github repo containing all the latest files and code