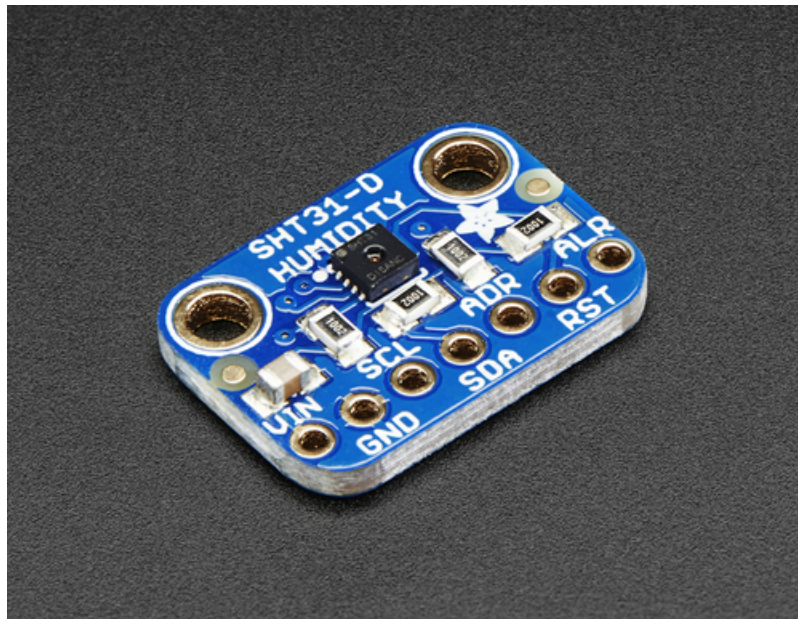


Adafruit SHT31-D Temperature & Humidity Sensor Breakout

Created by lady ada

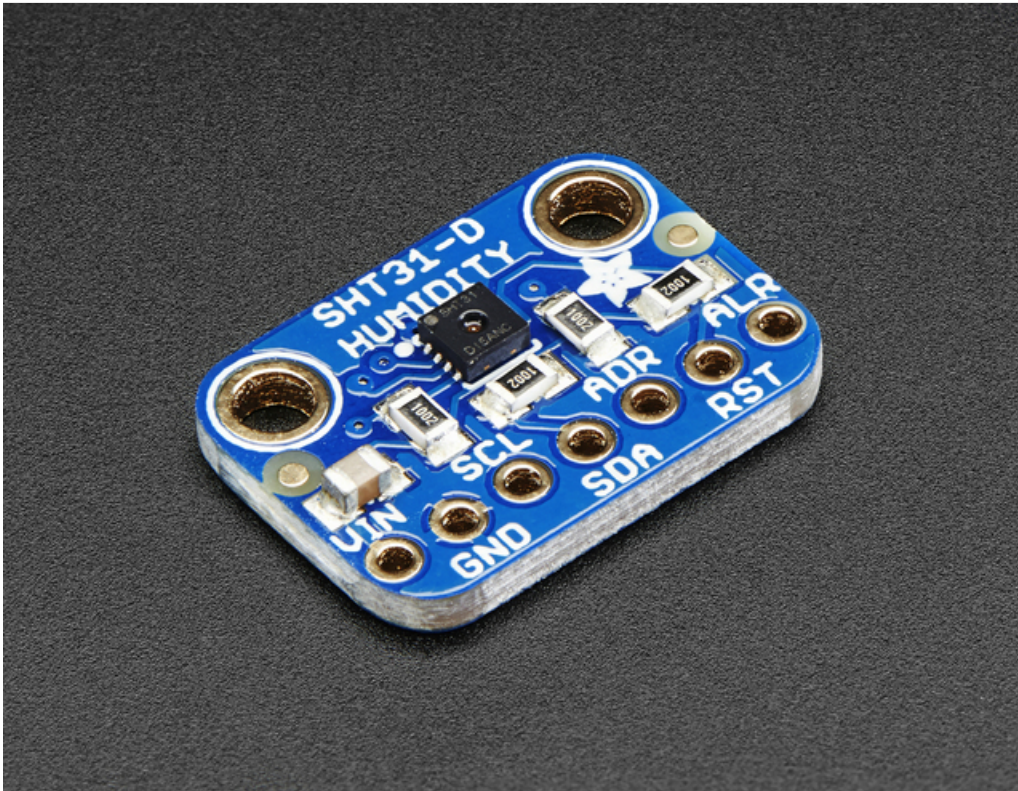


Last updated on 2018-03-07 05:34:42 PM UTC

Guide Contents

Guide Contents	2
Overview	3
Pinouts	6
Power Pins:	6
I2C Logic pins:	6
Other Pins:	6
Assembly	8
Prepare the header strip:	8
Add the breakout board:	9
And Solder!	10
Arduino Code	12
Download Adafruit_SHT31	12
Load Demo	13
Library Reference	14
CircuitPython Code	16
Usage	17
Downloads	19
Datasheets & Files	19
Schematic	19
Fabrication Print	19

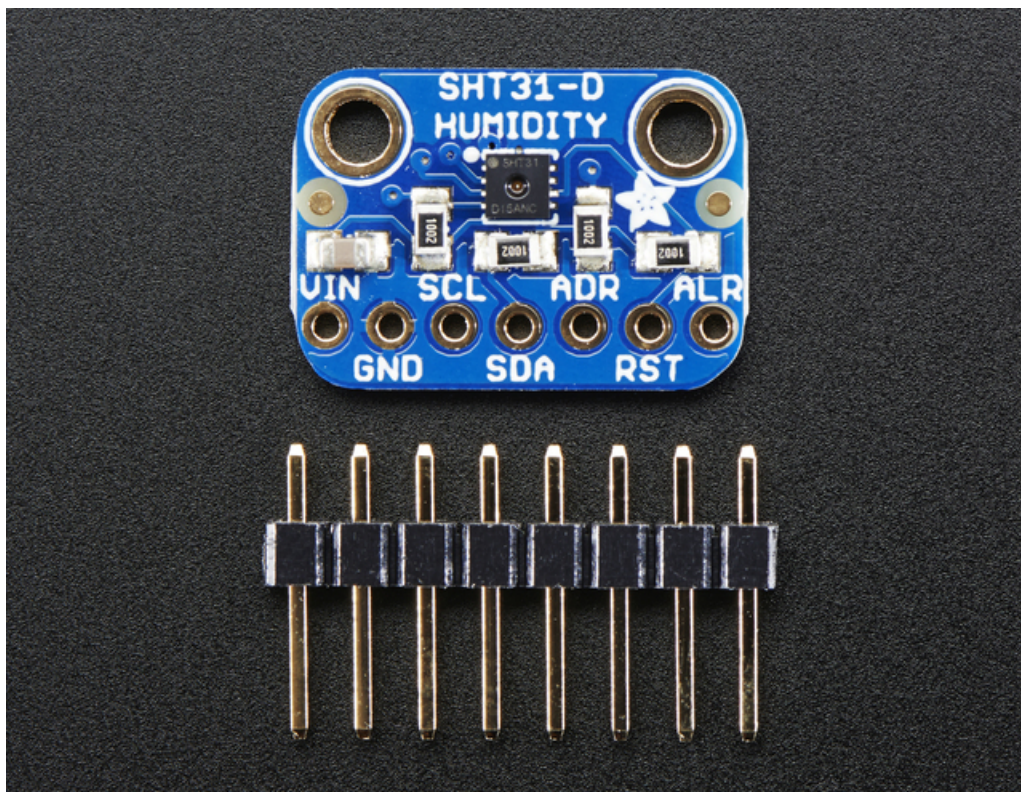
Overview



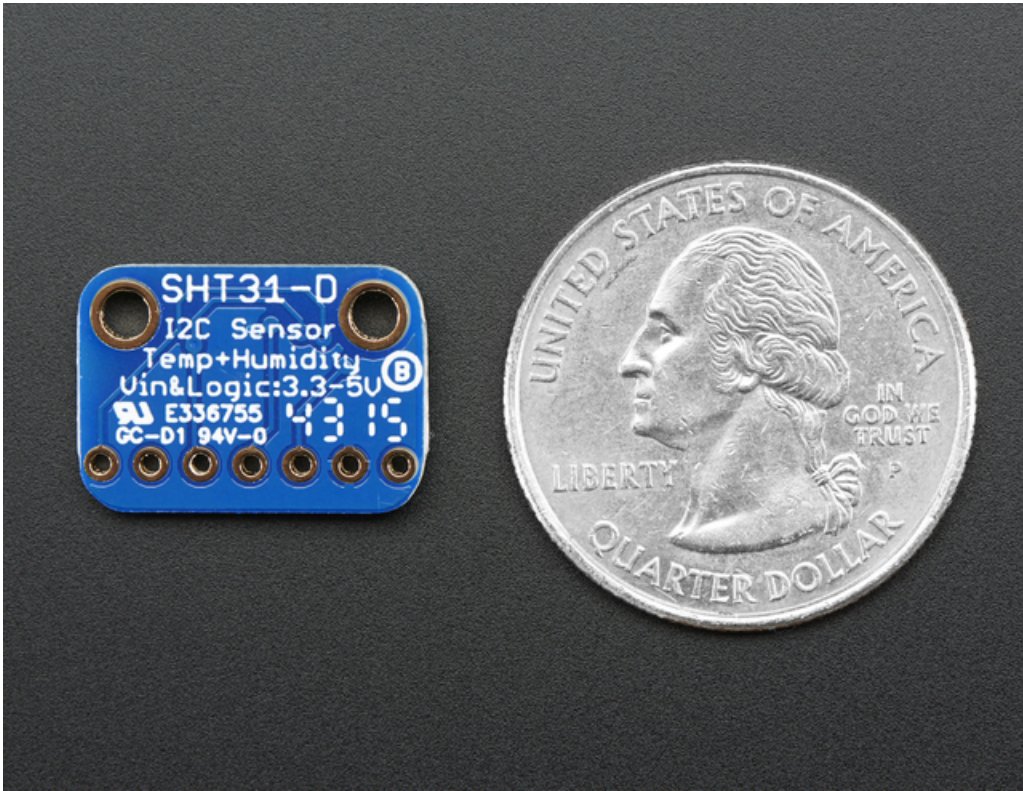
Sensirion Temperature/Humidity sensors are some of the finest & highest-accuracy devices you can get. And, *finally* we have some that have a true I2C interface for easy reading. The **SHT31-D** sensor has an excellent $\pm 2\%$ relative humidity and $\pm 0.3^\circ\text{C}$ accuracy for most uses.



Unlike earlier SHT sensors, this sensor has a *true* I2C interface, with two address options. It also is 3V or 5V compliant, so you can power and communicate with it using any microcontroller or microcomputer.



Such a lovely chip - so we spun up a breakout board with the SHT31-D and some supporting circuitry such as pullup resistors and capacitors. Each order comes with one fully assembled and tested PCB breakout and a small piece of header. You'll need to solder the header onto the PCB but it's fairly easy and takes only a few minutes even for a beginner.



Pinouts

The HTU21D-F is a I2C sensor. That means it uses the two I2C data/clock wires available on most microcontrollers, and can share those pins with other sensors as long as they don't have an address collision. For future reference, the default I2C address is **0x44** and you can also select address **0x45** by connecting the **ADDR** pin to a high voltage signal.



Power Pins:

- **Vin** - this is the power pin. The chip can use 2.5-5VDC for power. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V. For a 3.3V controller like a Raspberry Pi, connect to 3.3V
- **GND** - common ground for power and logic

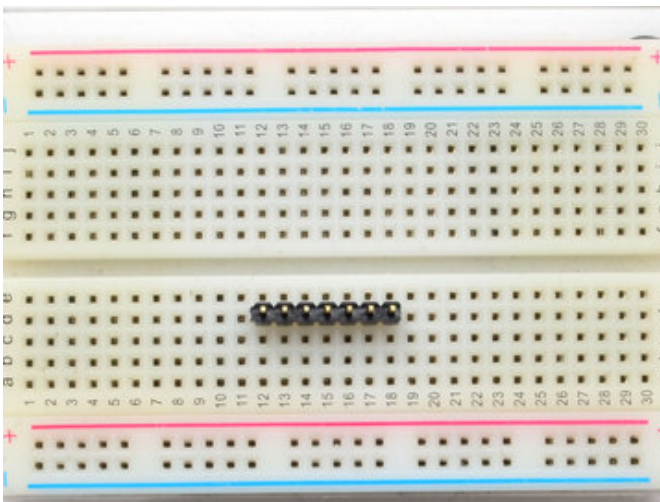
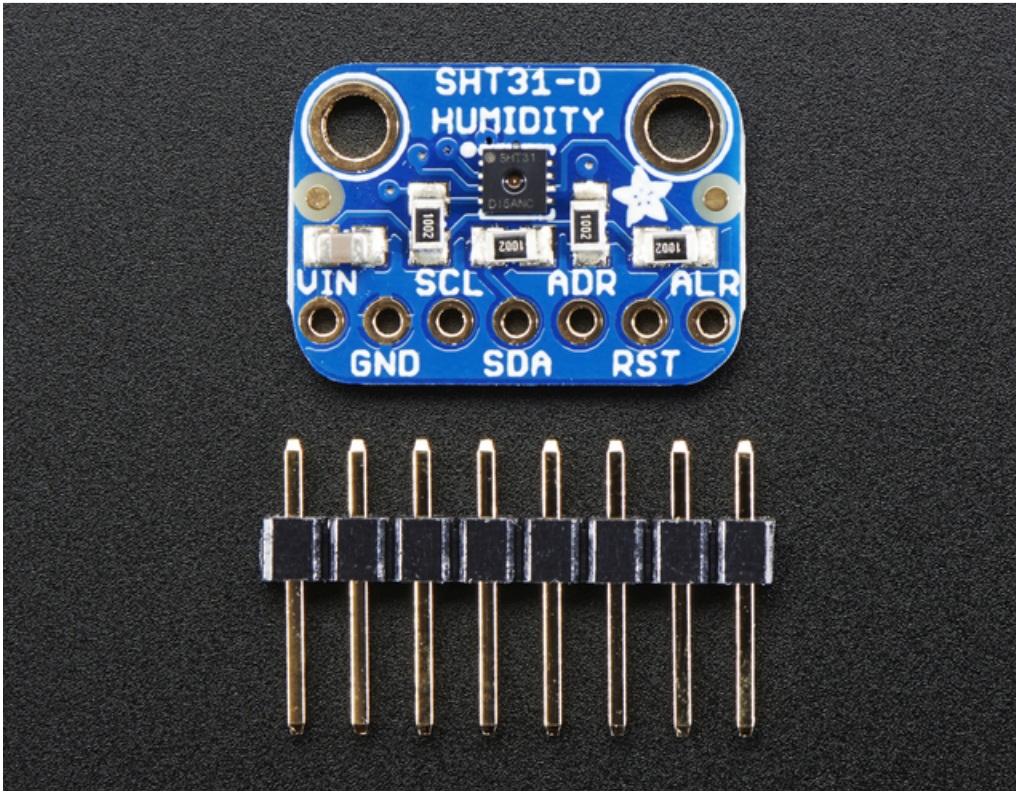
I2C Logic pins:

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. This pin has a 10K pullup resistor to Vin
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. This pin has a 10K pullup resistor to Vin

Other Pins:

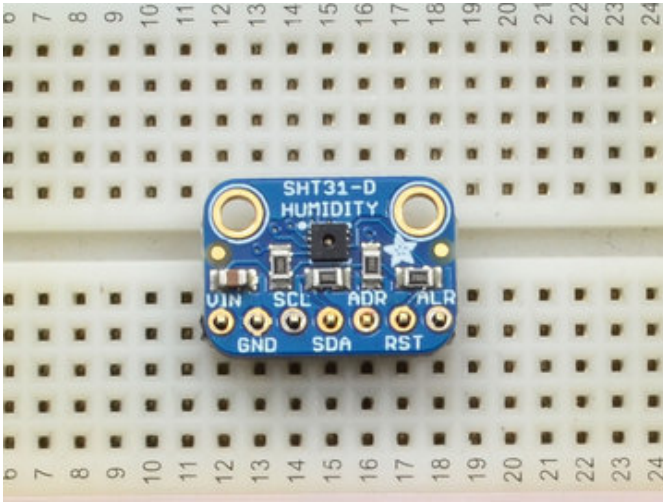
- **ADDR** - This is the I2C address selection pin. This pin has a 10K pull down resistor to make the default I2C address **0x44**. You can tie this pin to Vin to make the address **0x45**
- **RST** - Hardware reset pin. Has a 10K pullup on it to make the chip active by default. Connect to ground to do a hardware reset!
- **ALR** - Alert/Interrupt output. You can set up the sensor to alert you when an event has occurred. Check the datasheet for how you can set up the alerts

Assembly



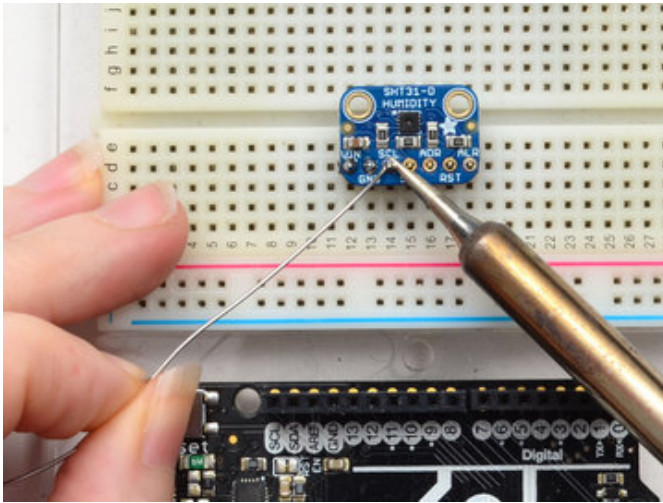
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



Add the breakout board:

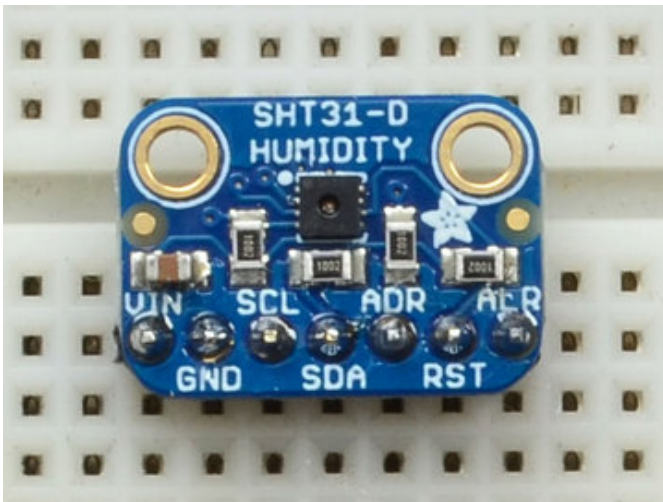
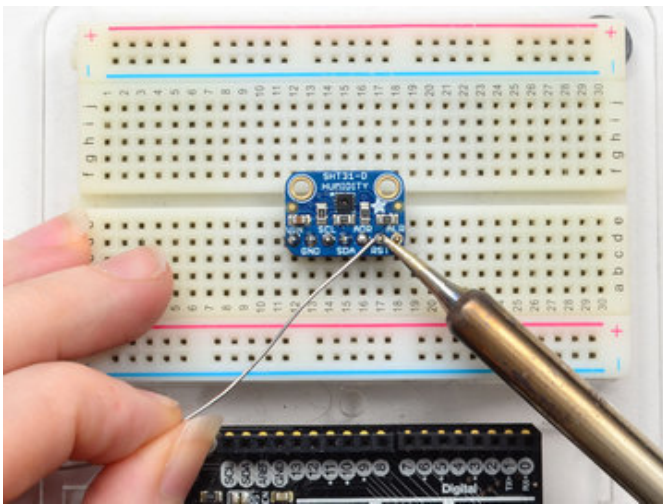
Place the breakout board over the pins so that the short pins poke through the breakout pads



And Solder!

Be sure to solder all pins for reliable electrical contact.

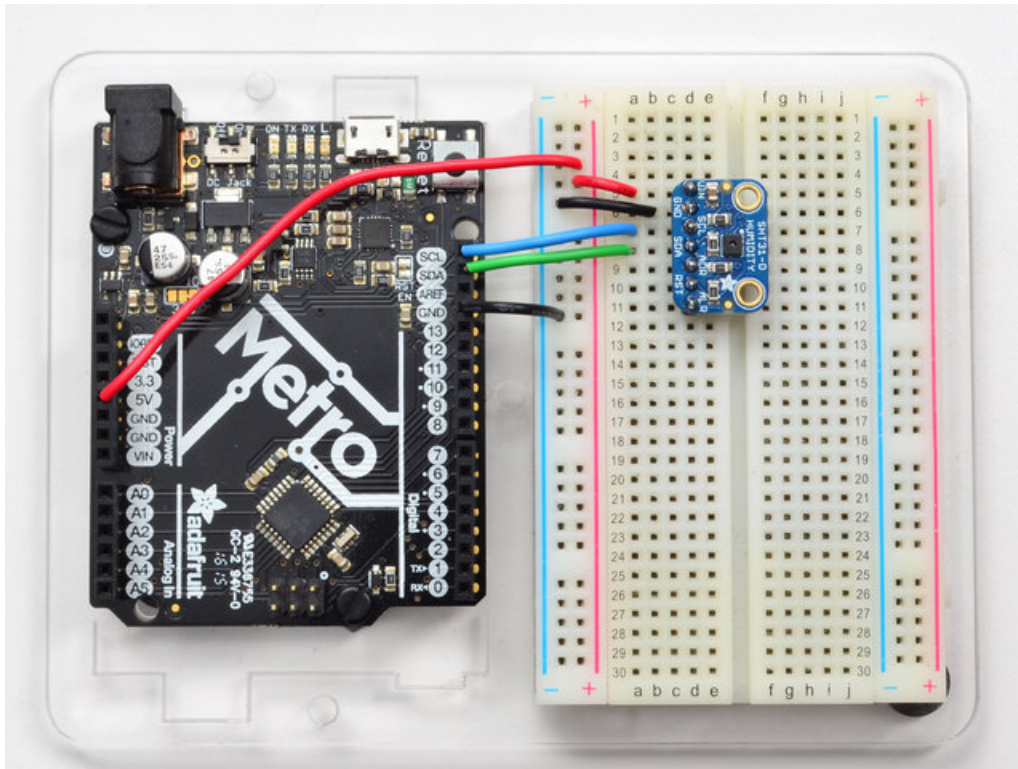
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](https://adafruit.it/aTk) (<https://adafruit.it/aTk>)).



You're done! Check your solder joints visually and continue onto the next steps

Arduino Code

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C, then port the code - its pretty simple stuff!



- Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The SHT31-D has a default I2C address of **0x44** which you can change to **0x45** by connecting the **ADR** pin to the **VIN** pin

Download Adafruit_SHT31

To begin reading sensor data, you will need to [download Adafruit_SHT31 from our github repository](#). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download Adafruit_SHT31 Library

<https://adafru.it/k6e>

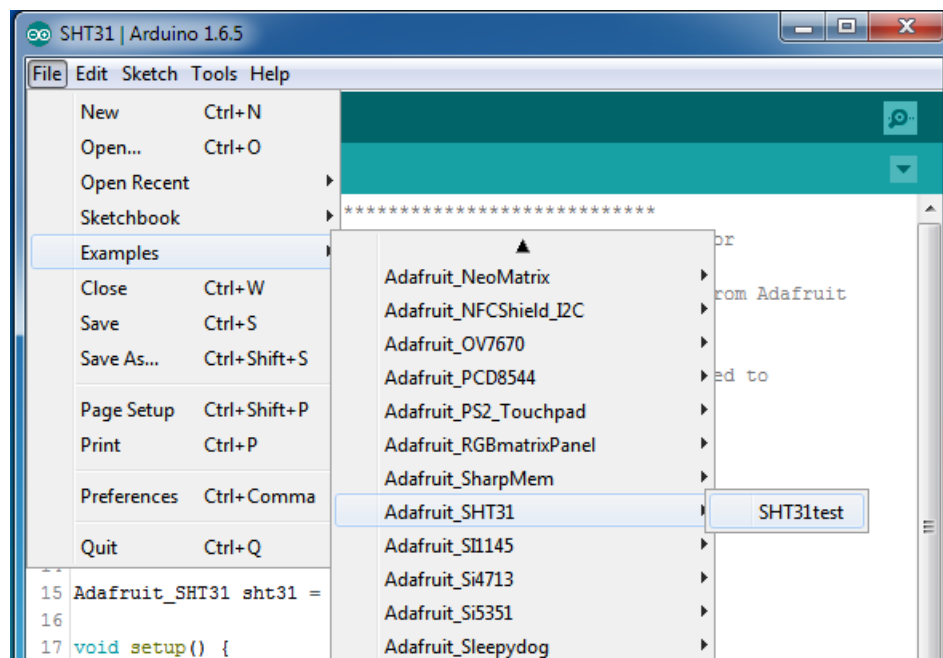
Rename the uncompressed folder **Adafruit_SHT31** and check that the **Adafruit_SHT31** folder contains **Adafruit_SHT31.cpp** and **Adafruit_SHT31.h**

Place the **Adafruit_SHT31** library folder your **arduinofolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

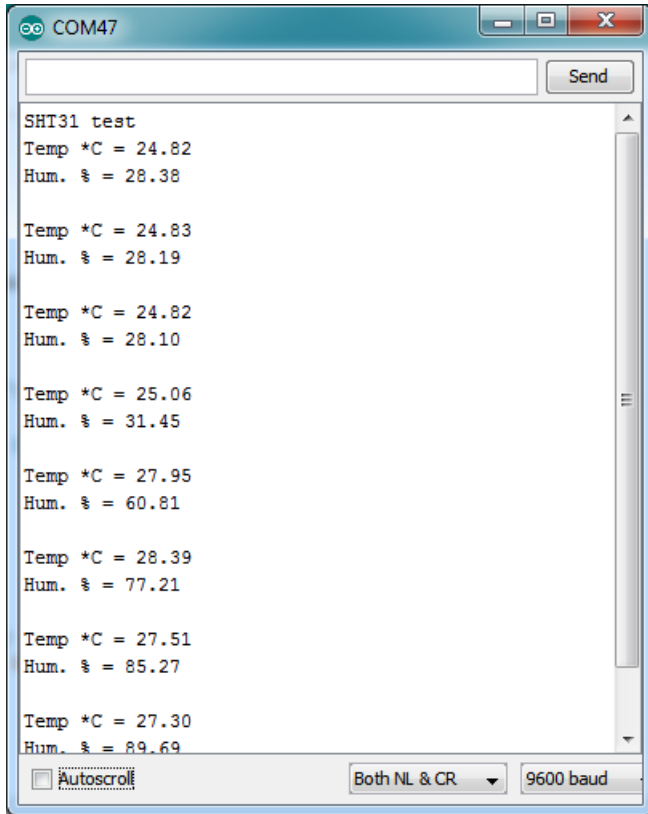
We also have a great tutorial on Arduino library installation at:
<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use>

Load Demo

Open up **File->Examples->Adafruit_SHT31->SHT31test** and upload to your Arduino wired up to the sensor



Thats it! Now open up the serial terminal window at 9600 speed to begin the test.



You can try breathing on the sensor to increase the humidity. The sensor reacts very fast!

Library Reference

The library we have is simple and easy to use

You can create the **Adafruit_SHT31** object with:

```
Adafruit_SHT31 sht31 = Adafruit_SHT31();
```

There are no pins to set since you must use the I2C bus!

Then initialize the sensor with:

```
sht31.begin(0x44)
```

This function returns **True** if the sensor was found and responded correctly and **False** if it was not found

The **0x44** is the i2c address you have the sensor set up for. By default its 0x44, you can also adjust the sensor for **0x45** and then pass that value in

Once initialized, you can query the temperature in °C with

```
sht31.readTemperature()
```

Which will return floating point (decimal + fractional) temperature. You can convert to Fahrenheit by multiplying by 1.8

and adding 32 as you have learned in grade school!

Reading the humidity is equally simple. Call

```
sht31.readHumidity()
```

to read the humidity also as a floating point value between 0 and 100 (this reads % humidity)

We also have a few helper functions. Want to soft-reset the sensor? Use

```
sht31.reset()
```

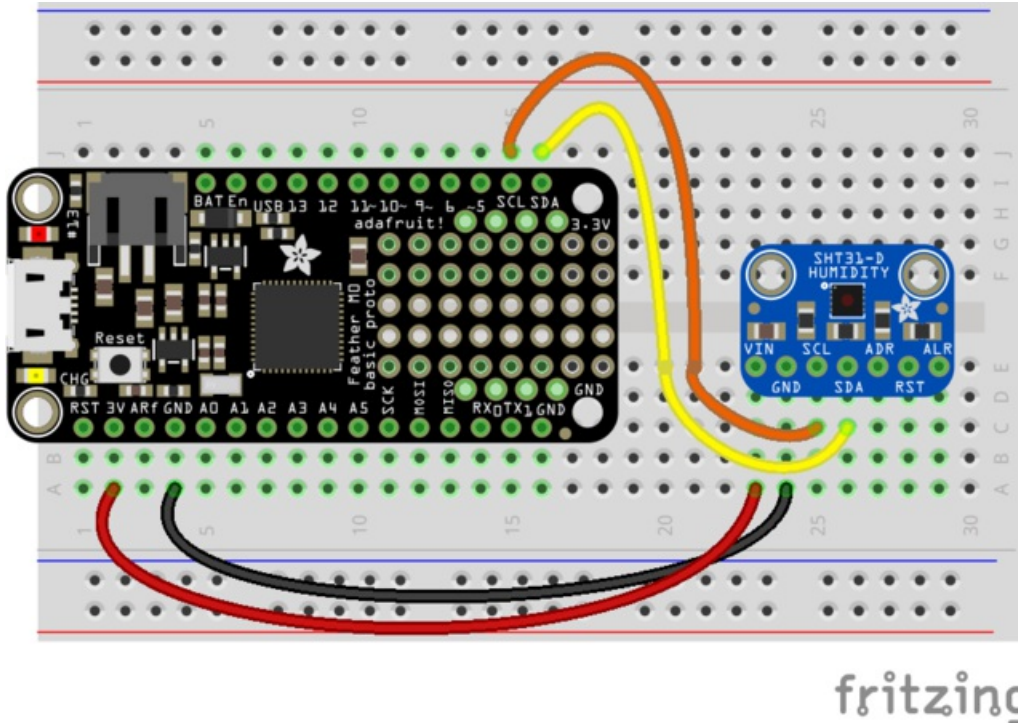
There's also a heater built into the sensor, used to heat/evaporate any condensation. You can turn it on or off with

```
sht31.heater(true)  
sht31.heater(false)
```

CircuitPython Code

It's easy to use the SHT31-D sensor with CircuitPython and the [Adafruit CircuitPython SHT31D](#) module. This module allows you to easily write Python code that reads the humidity and temperature from the sensor.

First wire up a SHT31-D to your board exactly as shown on the previous pages for Arduino using an I2C connection. Here's an example of wiring a Feather M0 to the sensor with I2C:



- Board 3V to sensor VIN
- Board GND to sensor GND
- Board SCL to sensor SCL
- Board SDA to sensor SDA

Next you'll need to install the [Adafruit CircuitPython SHT31D](#) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython](#) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle](#). Our introduction guide has [a great page on how to install the library bundle](#) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- `adafruit_sht31d.mpy`
- `adafruit_bus_device`

You can also download the `adafruit_sht31d.mpy` from [its releases page on Github](#).

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_sht31d.mpy`, and `adafruit_bus_device` files and folders copied over.

Next [connect to the board's serial REPL](#) so you are at the CircuitPython >>> prompt.

Usage

To demonstrate the usage of the sensor we'll initialize it and read the humidity and temperature from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
import adafruit_sht31
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_sht31.SHT31(i2c)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
import adafruit_sht31
i2c = bitbangio.I2C(board.SCL, board.SDA)
sensor = adafruit_sht31.SHT31(i2c)
```

Now you're ready to read values from the sensor using any of these properties:

- **relative_humidity** - The relative humidity measured by the sensor, this is a value from 0-100%.
- **temperature** - The temperature measured by the sensor, a value in degrees Celsius.

```
print('Humidity: {0}%'.format(sensor.relative_humidity))
print('Temperature: {0}C'.format(sensor.temperature))
```

```
>>> print('Humidity: {0}%'.format(sensor.relative_humidity))
Humidity: 38.8291%
>>> print('Temperature: {0}C'.format(sensor.temperature))
Temperature: 22.9545C
>>> █
```

That's all there is to using the SHT31D with CircuitPython!

Below is a complete example that measures the sensor readings and prints them every two seconds. Save this as **main.py** on your board and open the REPL to see the output. Remember to change to the **bitbangio** module if necessary for your board!


```

import time
import board
import busio
import adafruit_sht31

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_sht31.SHT31(i2c)

loopcount = 0
while True:
    print("\nTemperature: %0.1f C" % sensor.temperature)
    print("Humidity: %0.1f %" % sensor.relative_humidity)
    loopcount += 1
    time.sleep(2)
    # every 10 passes turn on the heater for 1 second
    if loopcount == 10:
        loopcount = 0
        sensor.heater = True
        print("Sensor Heater status =", sensor.heater)
        time.sleep(1)
        sensor.heater = False
        print("Sensor Heater status =", sensor.heater)

```

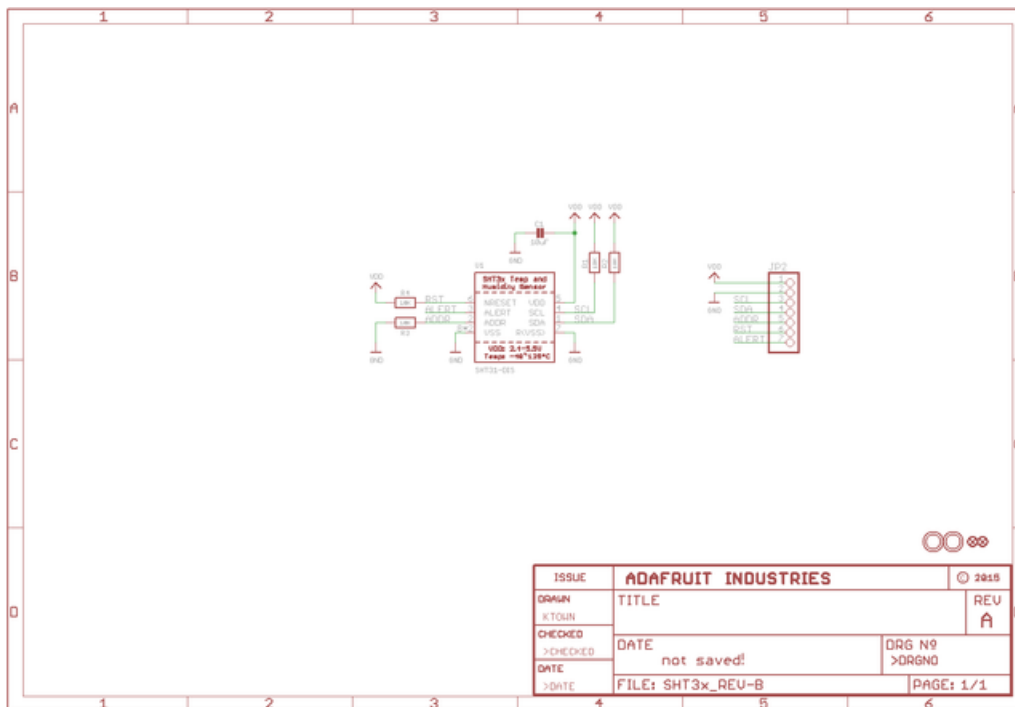
Downloads

Datasheets & Files

- [SHT31-DIS dataheet](#)
- [EagleCAD PCB Files on GitHub](#)
- [Fritzing object available in the Adafruit Fritzing Library](#)

Schematic

Click to enlarge



Fabrication Print

Dimensions in inches

