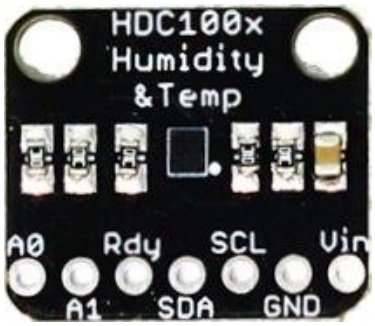
SmartElex HDC1008- Temperature & Humidity Sensor Breakout Board



This I2C digital humidity sensor is a fairly accurate and intelligent alternative to the much simpler Humidity and Temperature Sensor - SHT15. It has a typical accuracy of ±4% with an operating range that's optimized from 10% to 80% RH. Operation outside this range is still possible - just the accuracy might drop a bit. The temperature output has a typical accuracy of ±0.2°C from -20~85°C.

The HDC1008 sensor chip has 2 address-select pins, so you can have up to 4 shared on a single I2C bus. It's also 3-5V power and logic safe so you don't need any level shifters or regulators to use with a 5V or 3V microcontroller

This chip is only available in a tiny BGA package. So we created a breakout board with the chip and some extra passive components to make it easy to use. 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.

Please note: TI has indicated that there's a 'settling' effect for the humidity and that you will need to re-hydrate the sensor once you receive it. To rehydrate it, place it in a location with 85% humidity for 24 hours or 60% humidity for 10 days.

Pinouts

The HDC1008 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 **0x40** but you can adjust it by connecting the address pins to Vin ('high' logic votlage), for four possible addresess: **0x40, 0x41, 0x42** or **0x43**

Power Pins:

- **Vin** - this is the power pin. Unlike many sensors, this chip can be powered by 3-5 VDC, so there is no voltage regulator on board. Simply power the board with the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V. For a 3V ARM processor, use 3V
- **GND** - common ground for power and logic

I2C Logic pins:

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. 3-5V logic.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. 3-5V logic.

Optional Pins

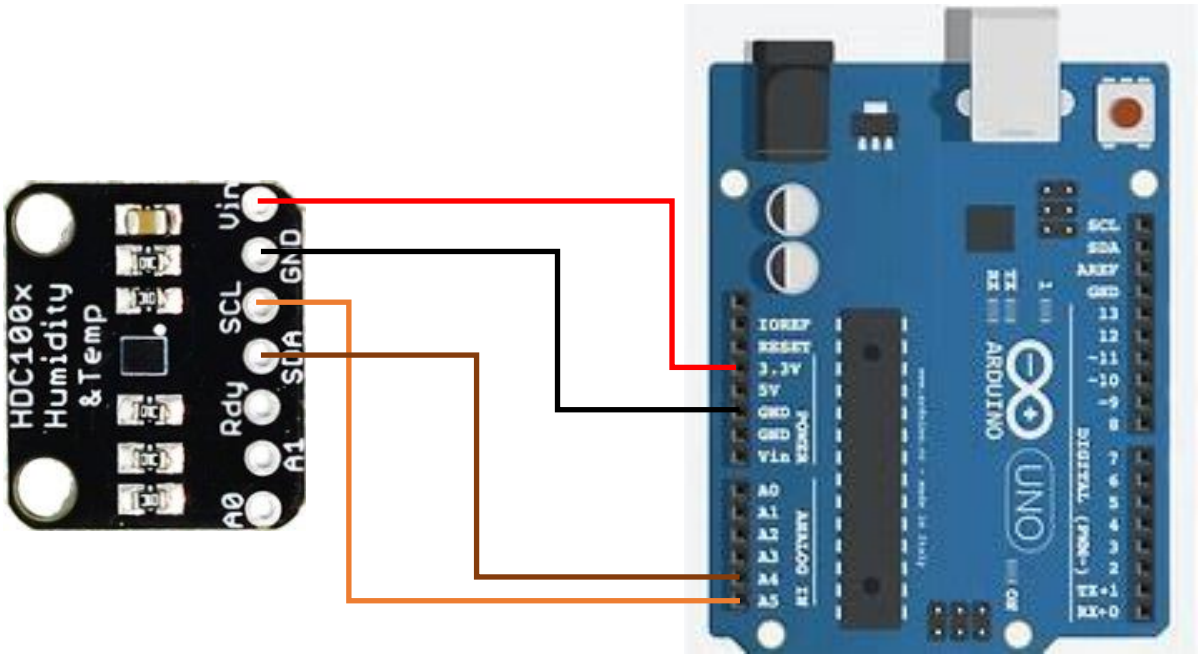
These are pins you don't need to connect to unless you want to! **RDY** - This is the interrupt/'ready' pin from the HDC100x. The chip has some capability to 'alert' you when data is ready to be read from the sensor. We don't use this pin in the library but it's available if you need it! It is **open collector** so you need to use a pull-up resistor if you want to read signal from this pin.

- **A0 A1** - These are the address select pins. Since you can only have one device with a given address on an i2c bus, there must be a way to adjust the address if you want to put more than one HDC100X on a shared i2c bus. The A0/A1 pins set the bottom 2 bits of the i2c address. There are pull-down resistors on the board so connect them to Vin to set the bits to '1'. They are read on power up, so de-power and re-power to reset the address

The default address is **0x40** and the address can be calculated by 'adding' the **A0/A1** to the base of 0x40
A0 sets the lowest bit with a value of **1**, **A1** sets the middle bit with a value of **2.** The final address is **0x40 + A1 + A0**.
So for example if **A1** is tied to Vin and **A0** is tied to Vin, the address is **0x40 + 2 + 1 = 0x43.**If only A0 is tied to Vin, the address is **0x40 + 1 = 0x41**
If only A1 is tied to Vin, the address is **0x42 + 2 = 0x42**

Wiring

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.



HDC1008	Arduino
SCL	SCL(A5)
SDA	SDA(A4)
VIN	5v OR 3.3v
GND	GND

- 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 HDC1008 has a default I2C address of **0x40**

Download Adafruit_HDC1000

To begin reading sensor data, you will need to download Adafruit_HDC1000 library from the Arduino library manager. Open up the Arduino library manager, Search for the **HDC1000** library and install it

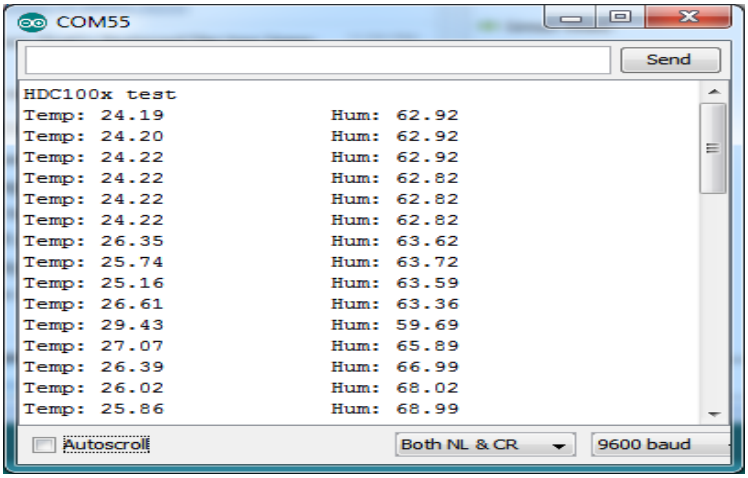
Load Example

Open up **File->Examples->HDC1000test** and upload to your Arduino wired up to the sensor

Example Code

```
#include "HDC1000.h"

HDC1000 hdc1000;

// Connect Vin to 3-5VDC
// Connect GND to ground
// Connect SCL to I2C clock pin (A5 on UNO)
// Connect SDA to I2C data pin (A4 on UNO)

void setup() {
  Serial.begin(9600);
  Serial.println("HDC100x test");

  if (!hdc.begin()) {
    Serial.println("Couldn't find sensor!");
    while (1);
  }
}
void loop() {
  Serial.print("Temp: "); Serial.print(hdc.readTemperature());
  Serial.print("\t\tHum: "); Serial.println(hdc.readHumidity());
  delay(500);
}
```

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



Please note: TI has indicated that there's a 'settling' effect for the humidity and that you will need to re-hydrate the sensor once you receive it. To rehydrate it, place it in a location with 85% humidity for 24 hours or 60% humidity for 10 days.

Library Reference

The library we have is simple and easy to use.
There are no pins to set since you must use the I2C bus!
Then initialize the sensor with:

```
hdc.begin()
```

if you aren't using the default 0x40 i2c address, you can pass in the i2c address to **begin** to have it use that one instead.

```
hdc.begin(0x42)
```

this function returns **True** if the sensor was found and responded correctly and **False** if it was not found
Once initialized, you can query the temperature in °C with

```
hdc.readTemperature()
```

Which will return floating point (decimal + fractional) temperature. You can convert to Fahrenheit by multiplying by 1.8 and adding 32.
Reading the humidity is equally simple. Call

```
hdc.readHumidity()
```

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