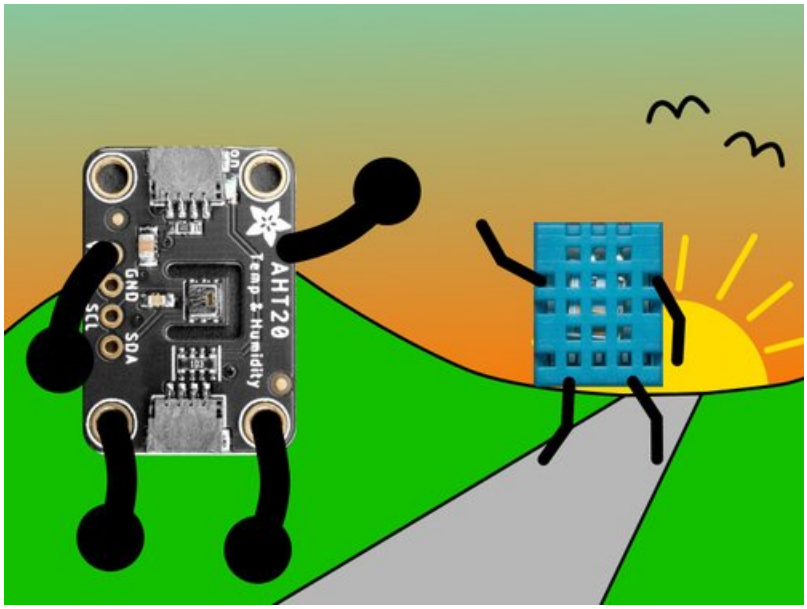
adafruit learning system

Modern Replacements for DHT11 and DHT22 Sensors

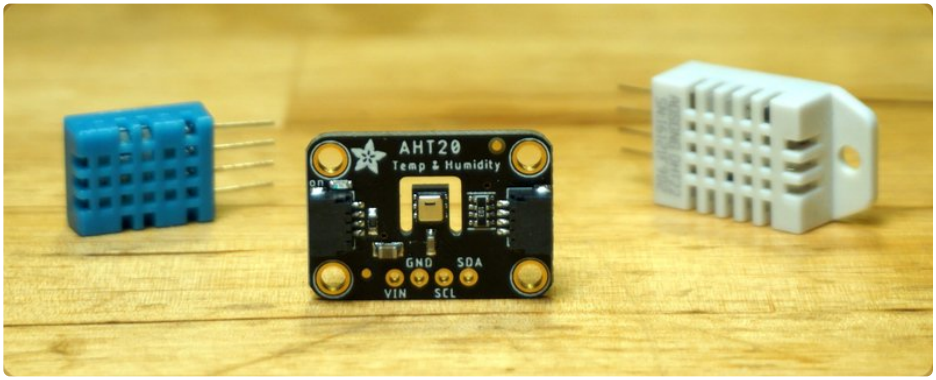
Created by Carter Nelson



Last updated on 2021-09-01 01:56:00 PM EDT

Guide Contents

Overview



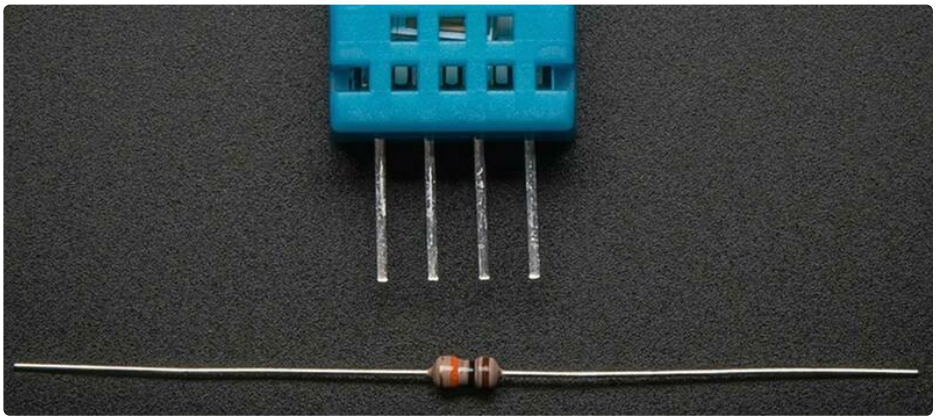
The appearance of the Arduino UNO (https://adafru.it/UBz), back in the 2010 time frame, really changed the landscape of easily programmable microcontrollers. The associated Arduino IDE, along with numerous libraries that were being published, made writing and uploading programs much easier (by a lot) than previously possible. Attaching various sensors, to measure things like pressure or acceleration, suddenly became very accessible.

When it came to measuring humidity, the DHT11 (https://adafru.it/f6N) and DHT22 (https://adafru.it/sb4) sensors dominated the market. These were super cheap, readily available, and had ready-to-go Arduino libraries. Tons of guides were published using these sensors as demos.

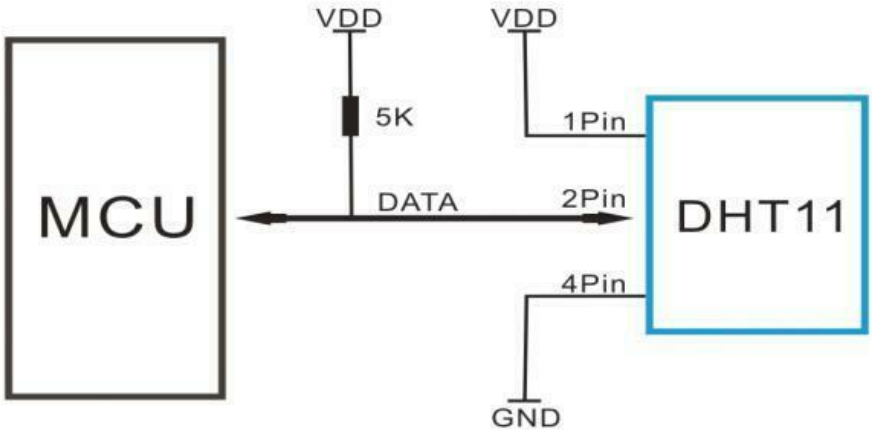
Fast forward to now. The microcontroller market has evolved significantly. While you can still buy an Arduino UNO (https://adafru.it/UBA), there are now much more powerful options available. Even powerful enough to allow running an interpretive language like CircuitPython (https://adafru.it/cpy-welcome). Also, single board computers, like the Raspberry Pi (https://adafru.it/OEx), which run a full Linux operating system, provide a totally different class of host controllers.

In this modern landscape, are the DHT11/22 still good cheap choices for measuring humidity? The answer is **NO**. In this guide we go into why that is by briefly looking at how the DHT11/22 sensors work. We then provide some information on modern alternatives that should generally be used instead of the DHT11/22.

Why are they problematic?



Even though these sensors typically have 4 pins, only 3 are used. For example, here is the wiring diagram from the DHT11 datasheet (https://adafru.it/aJY):



In the diagram above, "MCU" is the microcontroller, i.e your Arduino UNO, Feather M4 Express, Raspberry Pi, or whatever you are connecting the DHT11/22 sensors to. **There's only one pin used for data**. This is a digital pin, so is either HIGH or LOW. The sensor readings (humidity, temperature) need to be sent over as a stream of 1's and 0's. But how is that done? Sure, HIGH=1 and LOW=0, but how does the MCU know **when** to read the DATA line? The answer is timing.

Here is a timing diagram taken from the same datasheet:

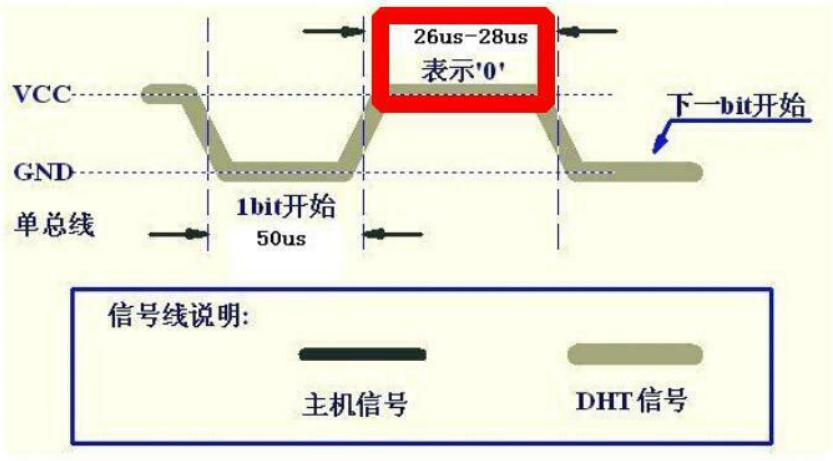
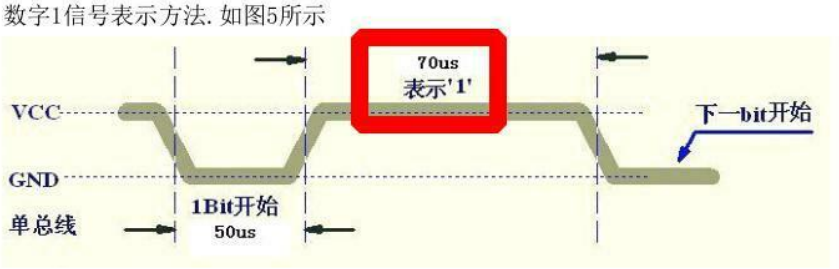


图4
数字1信号表示方法. 如图5所示



So a 0 is represented by a HIGH time of 26us-28us and a 1 is represented by a HIGH time of 70us. That "us" is units of **micro**seconds. To put that in some perspective, Arduino's much used `delay()` (https://adafru.it/COV) function uses units of **milli**seconds. The smallest delay you can program with that function is:

```
delay(1);
```

which ends up being **1000 microseconds**. Yes, there is the Arduino `delayMicroseconds()` function which works in units of microseconds. But delaying is not what is needed to read the signals. It's the ability to measure the pulse with microsecond level precision. Or at least with enough precision to be able to discern the 0 pulse from a 1 pulse. This can be non-trivial.

Trivial If...

There are definitely ways to measure that microsecond level data signal. If you had a microcontroller, like an Arduino UNO, and **all** it had to do was measure that signal, then it could do so reasonably well. Here's the section of code from the Adafruit DHT Arduino library (https://adafru.it/aJX) that does the actual timing measurement:

```
uint8_t portState = level ? _bit : 0;
  while ((*portInputRegister(_port) & _bit) == portState) {
    if (count++ >= _maxcycles) {
      return TIMEOUT; // Exceeded timeout, fail.
    }
  }
```

That code is for AVR microcontrollers (https://adafru.it/UBB), like the Arduino UNO. For non-AVR boards, like an ARM based (https://adafru.it/UBC) Feather M4 Express, this code is used:

```
while (digitalRead(_pin) == level) {
    if (count++ >= _maxcycles) {
      return TIMEOUT; // Exceeded timeout, fail.
    }
  }
```

Both of these do the same thing - loop forever counting clock cycles until the pin state changes. **Nothing else is happening while these loops run! And there are about 40 bits that need to be read!**

So. If you can spare all those clock cycles to **nothing** but reading the signal, it can be done.

Somewhat Trivial If...

Another approach is to use a timer/counter peripheral to do the heavy lifting. Instead of directly reading the data signal in code (like previous section), a separate dedicated piece of hardware can read the signal. Most microcontrollers provide one or more timer/counter.

However, this peripheral still needs to be configured and managed by the host microcontroller. Doing that to read in all the bits from the DHT11/22 can end up with the same issue as above - trivial if you have nothing else to do.

Non-Trivial If...

Once you move beyond a simple "hello world" example of reading and printing the DHT11/22 senor values, things can quickly become non-trivial. The blocking (nothing else can run) nature of the data read can get in the way of other tasks.

Another example is when trying to use a DHT11/22 on a single board computer like a Raspberry Pi. Now there is a full Linux operating system running numerous processes. The approach of simply "do nothing but read data pulses" does not work well (if at all) on such a system.

Even If...

These sensors also tend to be just plain finicky. Even if the tight timing requirement can be met,

sometimes the signal read cannot be decoded properly.

That's why you see code like this, from the Adafruit DHT Arduino library example (https://adafru.it/UBD):

```
// Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
```

Basically - something happened, don't care what, just try again.

As a result...

The end result of all this is, well, take a look:

https://adafru.it/UBE

https://adafru.it/UBE

https://adafru.it/UBF

https://adafru.it/UBF

Maybe there are code and other fixes for all of these issues, but a better option is to just get away from this style sensor.

What are better alternatives?

Infomercial Announcer: The bit banging. The 100% CPU load. The corrupt readings. The incorrect readings...

Desperate DHT11/22 User: **There's got to be a better way !!!**

OK, so what are better alternatives? In general, anything that uses a clock signal. This could be SPI, but most sensors of this class will end up using I2C. Not only does the clock signal eliminate the need for tight timing requirements, most microcontrollers have dedicated I2C peripherals for easy interfacing.

The DHT11/22 sensors have two primary measurements:

- Humidity
- Temperature

These are basically humidity sensors, temperature just comes along for the ride. So take a look at all available humidity sensor options here:

https://adafru.it/UBG

https://adafru.it/UBG

If you have specific requirements in terms of power or accuracy, then take a look at each one's datasheets for specifications.

From The Makers Of...

OK, you want something as cheap as those ubiquitous DHT11/22 sensors? Well, from the same folks that brought you those, now comes the AHT20 sensor. This is a modern spin which uses I2C. And they cost like 5 bucks. Neat!

And it comes in various packaging options. There's a breakout version with STEMMA QT connectors:

Adafruit AHT20 - Temperature & Humidity Sensor Breakout Board

The AHT20 is a nice but inexpensive temperature and humidity sensor from the same folks that brought us the DHT22. You can take...

$4.50

In Stock

Add to Cart

Another version with pig tail wires:

AM2301B - Wired Enclosed AHT20 - Temperature and Humidity Sensor

The AM2301B is a nice but inexpensive temperature and humidity sensor

$7.95

In Stock

Add to Cart

And for that old school look, a version in the classic DHT11/22 waffle brick style:

DHT20 - AHT20 Pin Module - I2C Temperature and Humidity Sensor

The DHT20 is a nice but inexpensive temperature and humidity sensor from the same folks that brought us the DHT22.This little...

$4.50

In Stock

Add to Cart

So cost can no longer be an excuse for choosing a DHT11/22.