# DHT Simple Library

## Overview

I generated this library because I could not find a DHT library that worked on all my applications. They either were not reliable, interfered with another library or caused my VGA monitor to blink every time the sensor was read. This library uses only Arduino functions and is very easy to use and understand.

This library works with all known DHT sensors and should work ESP32 or ESP8266 processor boards. I tested it with ESP32 WROOM Dev Board, ESP32 D1 R32 and ESP8266 D1 R2 boards. Tested with DHT11 and DHT22 (AM2302) sensors.

## Examples

DHTSimple DHT1(DHT1PIN, DHTSensorType); //Constructor

The constructor generates an object named DHT1 with data pin DHT1PIN and sensor type DHTSensorType. DHT11 sensors are type 1 and all others are type 2 (DHT21, DHT22, AM2301, AM2302). The constructor associates the data pin and sensor type with object DHT1.

DHT1.begin();

This sets the data pin to OUTPUT and sets the line high to start the DHT sensor.

String s = DHT1.getData(h, t);

This reads data from the DHT sensor. String s has the status and float variables h and t contain the humidity and temperature readings. The humidity is relative humidity for 0 to 100 percent. The temperature is in degrees Celsius from -40 to 80. The String s values are as follows:

1. “Sensor Bad” – This occurs when the Sensor did not pull down the data line. It normally occurs when the data pin is incorrect, the sensor is not powered or not connected correctly.
2. “No Start Bit” – This indicates the Sensor did not pull up the data line. If the wire is long a pull-up resistor may be required.
3. “Start Bit Error” – This indicates the start bit was too short or the too long.
4. “Data Read Error” – This indicates the sensor stopped outputting data.
5. “No Data” – This indicates that all the data was 0 including the checksum.
6. Checksum Error – This indicates that the checksum value is incorrect. This could occur if an interrupt happened while reading the sensor.
7. “OK” – The read was successful, and all checks passed. High confidence the readings are correct.

String s = DHT1.getDataRetry(h, t);

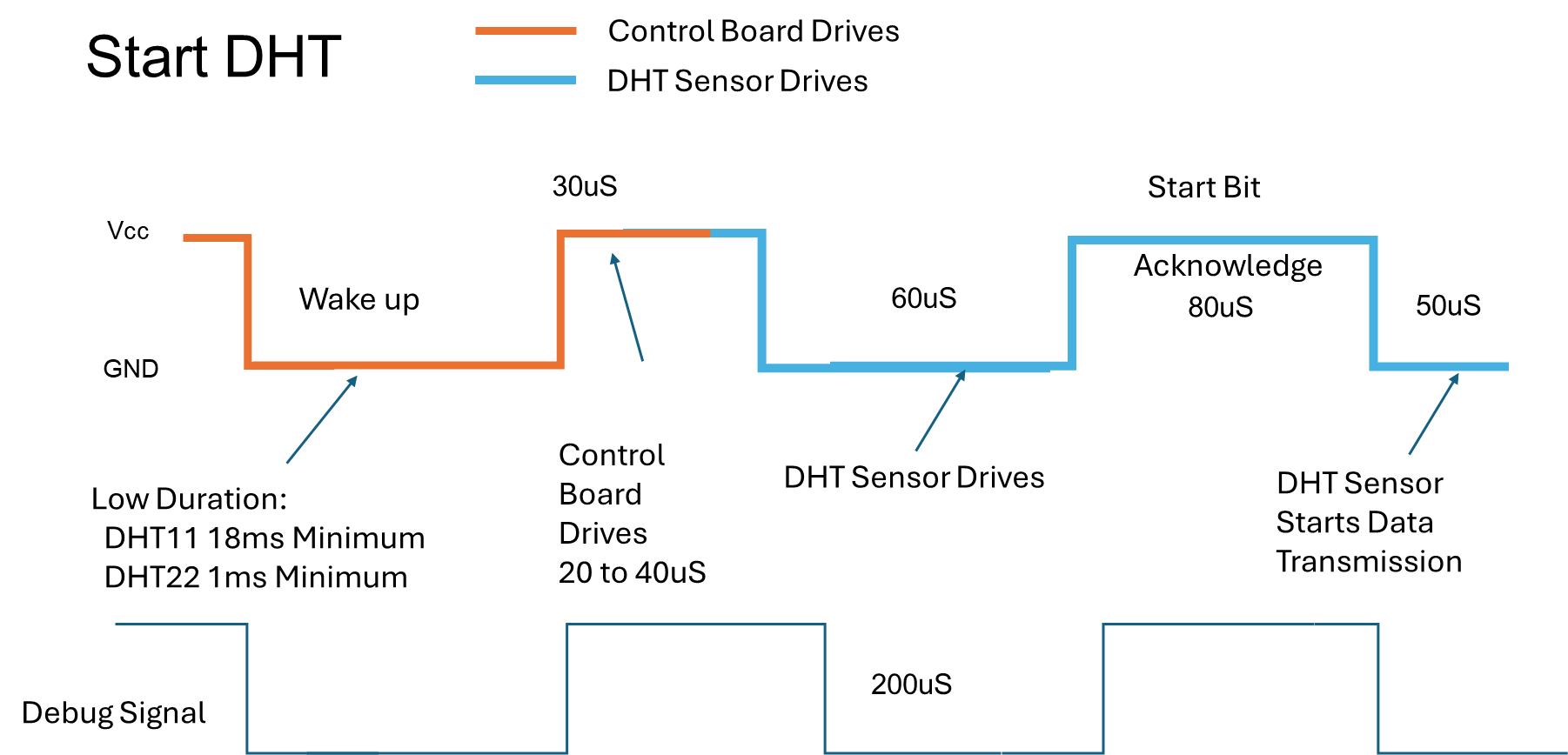
This line is the same as DHT1.getData except that the sensor is read three times if an error occurs. If there is an error three times in a row the error is returned in the s variable. If the program uses interrupts this is the recommended method to read the sensor.

DHT1.debug(DebugPin);

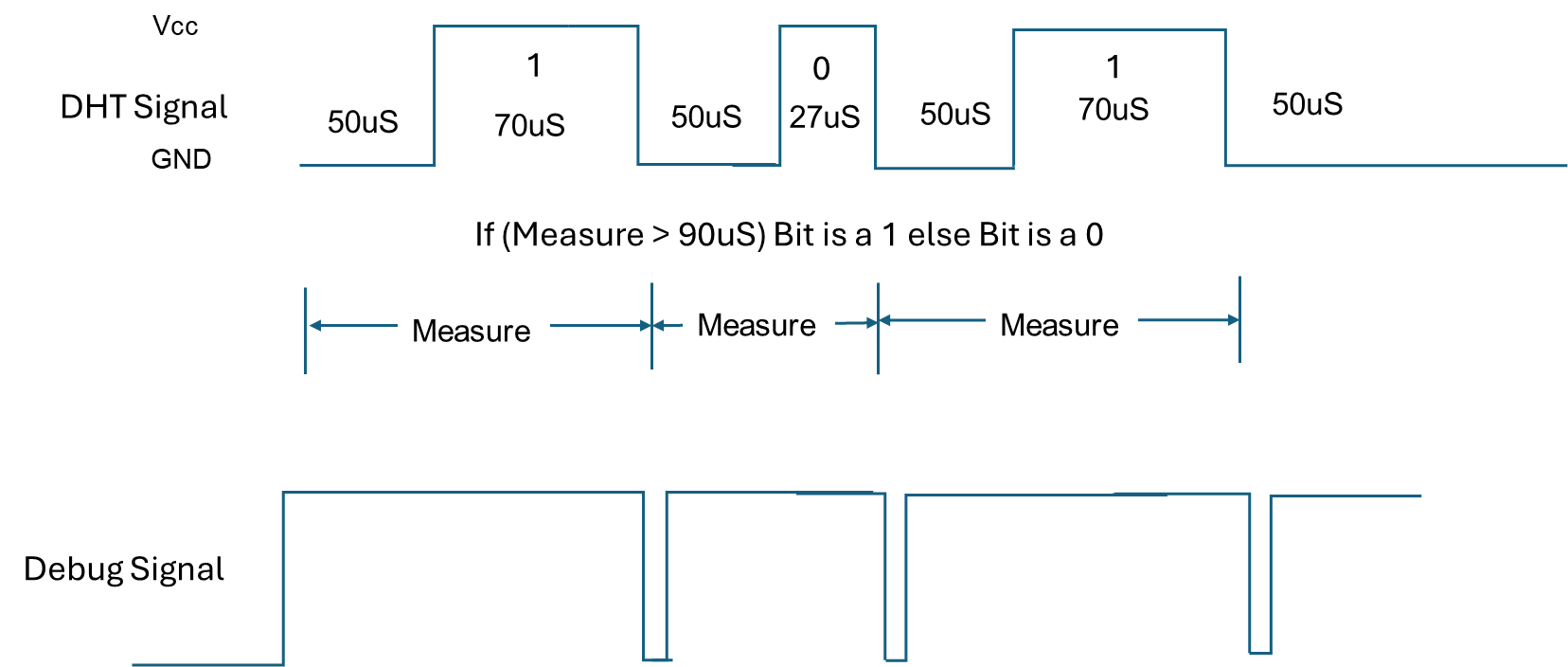
This instruction turns on an output pin to debug the sensor with a logic analyzer. The debug instruction must be before the begin instruction. The debug pin is shown in the Theory of Operation section.

## Theory of Operation

The data line is kept high until a read sensor operation is started. The output pin is pulled down to wake up the sensor. The duration of the LOW time depends upon the type of sensor. After the wakeup time the line is pulled HIGH for 20 to 40 microseconds, then the pin is put in INPUT\_PULLUP mode. The sensor then starts driving the line by pulling the pin low for approximately 60 microseconds. The sensor then pulls the line HIGH for 80 microseconds. This is the Start Bit.

[](image1.png)

After the Start Bit the data starts. When the data bit is a ‘1’ the data line is pulled high for 70 microseconds. When the data bit is a ‘0’ the data line is pulled high for 27 microseconds. The data line is pulled low for 50 microseconds between all the bits. The software measures the time from the low-to-low bits to determine the bit value. If the time is greater than 100 microseconds the bit is a ‘1’ and if it is less than 100 microseconds it is a ‘0’. The following figure illustrates reading the data.

[](image2.png)

## Other DHT Libraries

dhtESP32-rmt – Good library and easy to use. Did not seem to be compatible with the ESPAsyncWebServer library.

DHT\_sensor\_library – Another good library, but caused my display to blink every time the sensors were read.