DHT11

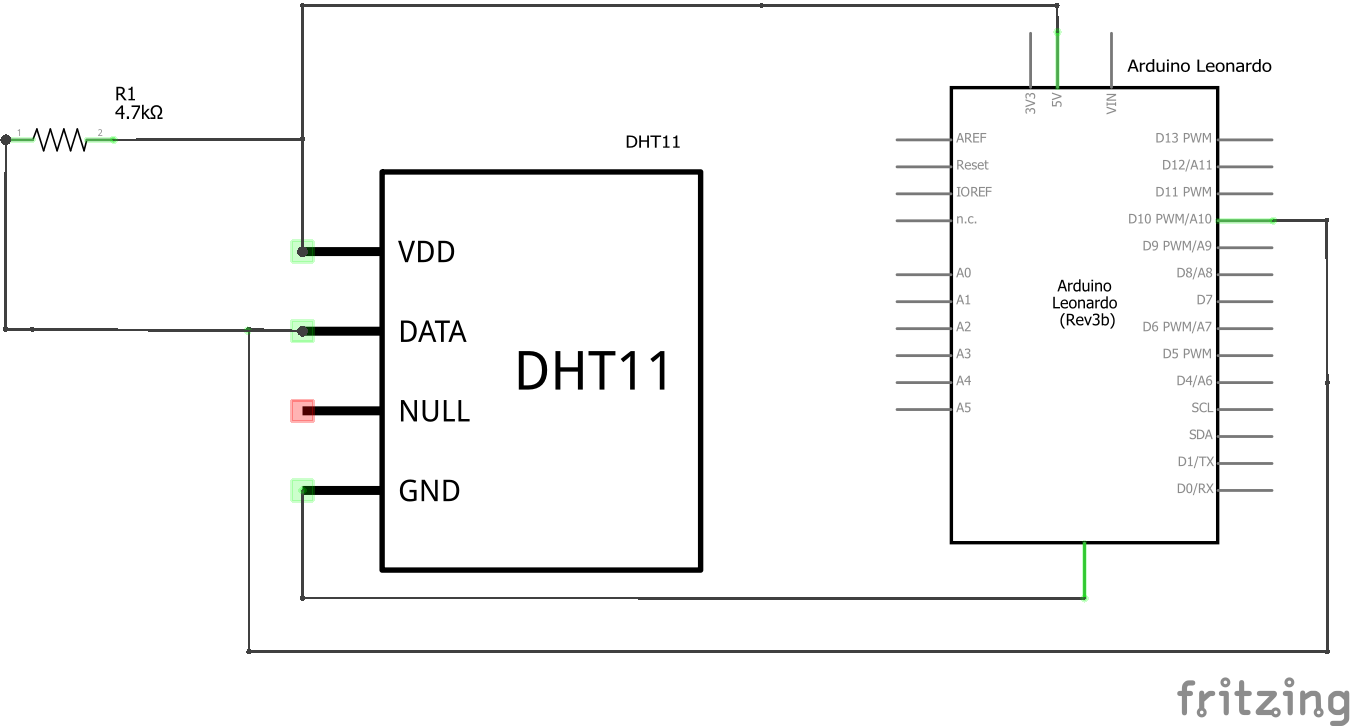
The DHT11 component is a temperature and humidity sensor.

The temperature is sensed using a thermistor component whose resistance value changes according to the temperature it is exposed to. The resistance is then measured by passing direct current through it and recording the voltage drop produced. The component is in fact an electrical resistor that is additionally sensitive to temperature.

The relative humidity is sensed using a capacitive humidity sensor. A small capacitor composed of hygroscopic dielectric material is placed between a pair of electrodes. At normal room temperature the dielectric constant of water vapor is about 80, much larger than the constant of the dielectric material (ranging from 2 to 15). Therefore, the absorption of moisture by the sensor results in increased sensor capacitance and relative humidity, which depends on ambient temperature and water vapor pressure, can be calculated using the relationship there exists with the amount of moisture in the sensor and sensor capacitance.

This sensor has a quite limited accuracy. The humidity can be measured only in the 20% to 90% range with an accuracy of while the temperature has a range of 0°C to 50°C and an accuracy of °C. These values are nonetheless acceptable for a consumer-grade alarm clock application. The sensor also has a sampling rate of 1Hz, so it cannot be polled more than once per second.

The following schematic shows how the component is wired up in the project. A pullup resistor of is applied to the DATA pin as the datasheet requires.



The sensor uses a serial interface called Single-Wire Two-Way. A single data line is used to send commands and to read responses. The whole communication process takes about 4ms.

The sensor sends the data in the following format:

|  |  |
| --- | --- |
| 8 bit | Integral part of humidity |
| 8 bit | Decimal part of humidity |
| 8 bit | Integral part of temperature |
| 8 bit | Decimal part of temperature |
| 8 bit | Checksum |

The DHT11 sensor is in low power mode unless it is transmitting a message. The MCU signals the sensor that he wants to read the data by sending a LOW voltage level on the DATA line, which when inactive is HIGH thanks to the pullup resistor, that must last at least 18ms. Then, the MCU must wait 20-40 s for the sensor’s response. The DHT response is an 80s LOW signal followed by an 80s HIGH signal. The data transmission then begins. Every bit is preceded by a 50s LOW signal, then the length of the following HIGH signals determines whether the bit is “0” or “1”. A 26-28 s signal is a “0” while a 70s signal is a “1”. At the end of the transmission the sensor sets the voltage LOW for 50s, then the pullup resistor will reset it to HIGH and the channel is free again.

The author of the SimpleDHT library we use in the project found out that in practice the aforementioned timings aren’t accurate, so in his code they are different. The original timings are still specified as comments.

In order to read from pin 11, the Arduino library performs the following operation:

Value = PINB & (1<<PINB7)

All pins are input by default, so we only need to read from the register PINB the value in position PINB7. In this case the value variable will have 0 in case of a LOW signal and a non-zero value (128) in case of a HIGH signal.

In our project, we read from the sensor relatively seldom, a rate of just one time per minute. The reason is twofold: one reason is that reading from the sensor implies using short delays (unless we wanted to write our own library that is) and these are to be avoided as much as possible, because the MCU won’t be responsive during the delay; the other reason is that there is no point in reading the values so often on a consumer-grade product when, unless solicited on purpose, changes in temperature and humidity happen so slowly.

//add finalized code details here

<https://www.ametherm.com/thermistor/what-is-an-ntc-thermistor>

<https://www.rotronic.com/en/humidity_measurement-feuchtemessung-mesure_de_l_humidite/capacitive-sensors-technical-notes-mr>