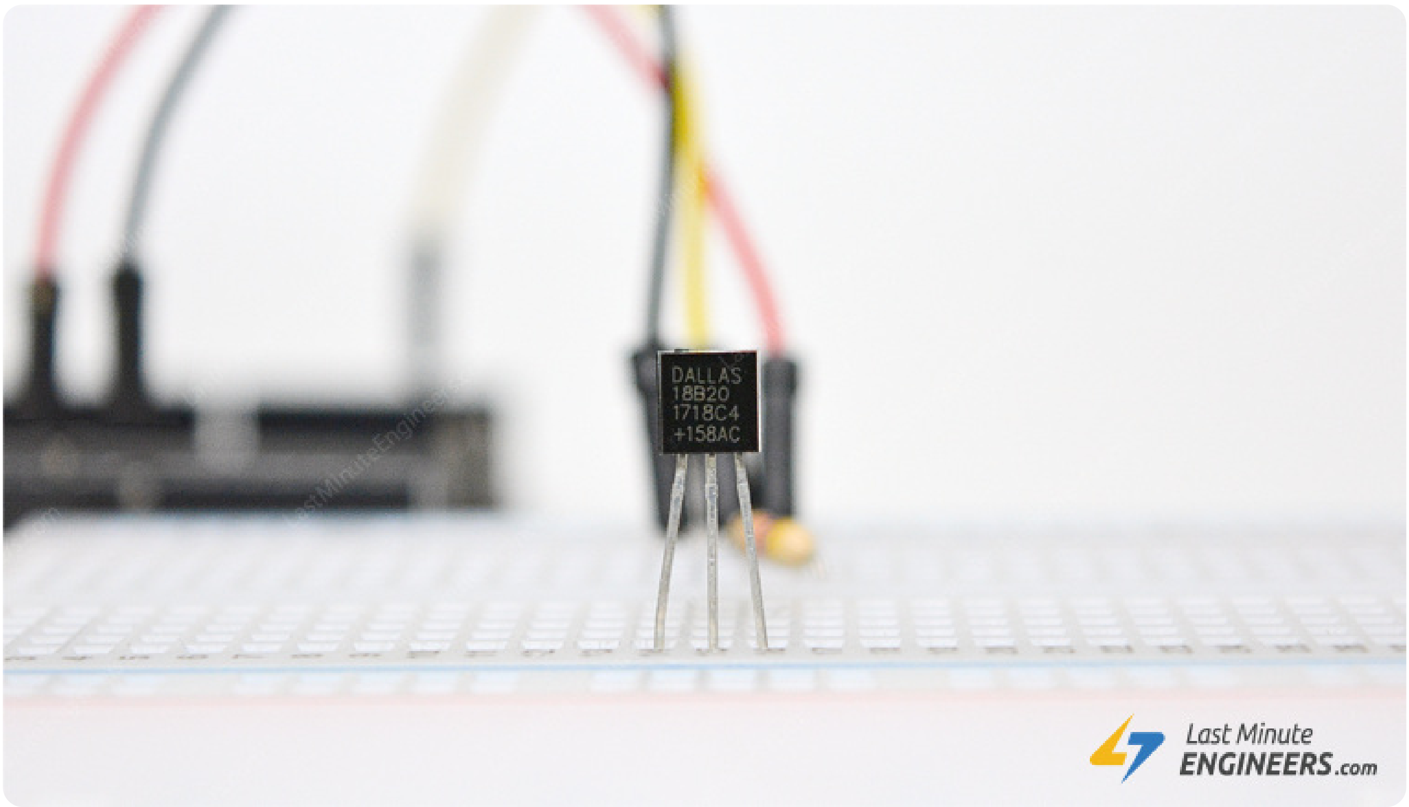
Interfacing DS18B20 1-Wire Digital Temperature Sensor with Arduino



One of the easiest and inexpensive way to add temperature sensing in your Arduino project is to use DS18B20 1-Wire Temperature Sensor. These sensors are fairly precise and needs no external components to work. So, with just a few connections and some Arduino code you'll be sensing temperature in no time!

DS18B20 1-Wire Temperature Sensor

DS18B20 is 1-Wire interface Temperature sensor manufactured by **Dallas Semiconductor Corp**. The unique 1-Wire® Interface requires only one digital pin for two way communication with a microcontroller.

The sensor comes usually in **two form factors**. One that comes in TO-92 package looks exactly like an ordinary transistor. Other one in a waterproof probe style which can be

Types Of DS18B20 Temperature Sensor

DS18B20 temperature sensor is fairly precise and needs no external components to work. It can measure temperatures from **-55°C to +125°C** with ±0.5°C Accuracy.

The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits. However, the default resolution at power-up is 12-bit (i.e. 0.0625°C precision).

The sensor can be powered with a 3V to 5.5V power supply and consumes only 1mA during active temperature conversions.

Here are the complete specifications:

Power Supply	3V to 5.5V
Current Consumption	1mA
Temperature Range	-55 to 125°C
Accuracy	±0.5°C
Resolution	9 to 12 bit (selectable)
Conversion Time	< 750ms

DS18B20 Temperature Sensor Specifications

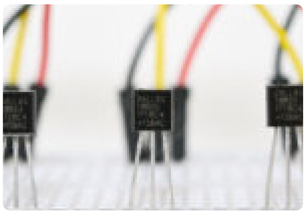
DS18B20 Datasheet

Multiple DS18B20 On Single Bus

One of the biggest advantages of DS18B20 is that multiple DS18B20s can coexist on the same 1-Wire bus. As each DS18B20 has a unique **64-bit serial code** burned in at the factory, it's easier to differentiate them from one another.

This feature can be a huge advantage when you want to control many DS18B20s distributed over a large area.

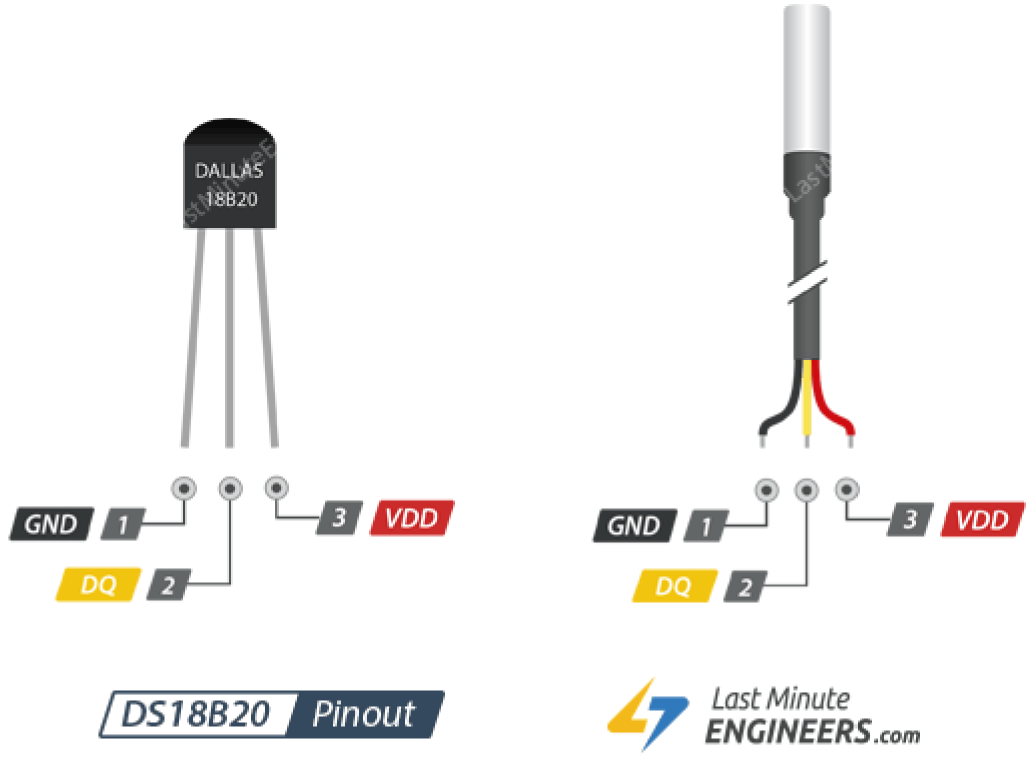
To know how to read temperature from multiple DS18B20 temperature sensors, please check this tutorial out.



Interfacing Multiple DS18B20 Digital Temperature Sensors with Arduino

One of the biggest advantages of DS18B20 is that multiple DS18B20 can coexist on the same 1-Wire bus. As each DS18B20 has a unique 64-bit serial...

DS18B20 Sensor Pinout



DQ is 1-Wire Data Bus should be connected to a digital pin on microcontroller.

VDD pin supplies power for the sensor which can be between 3.3 to 5V.

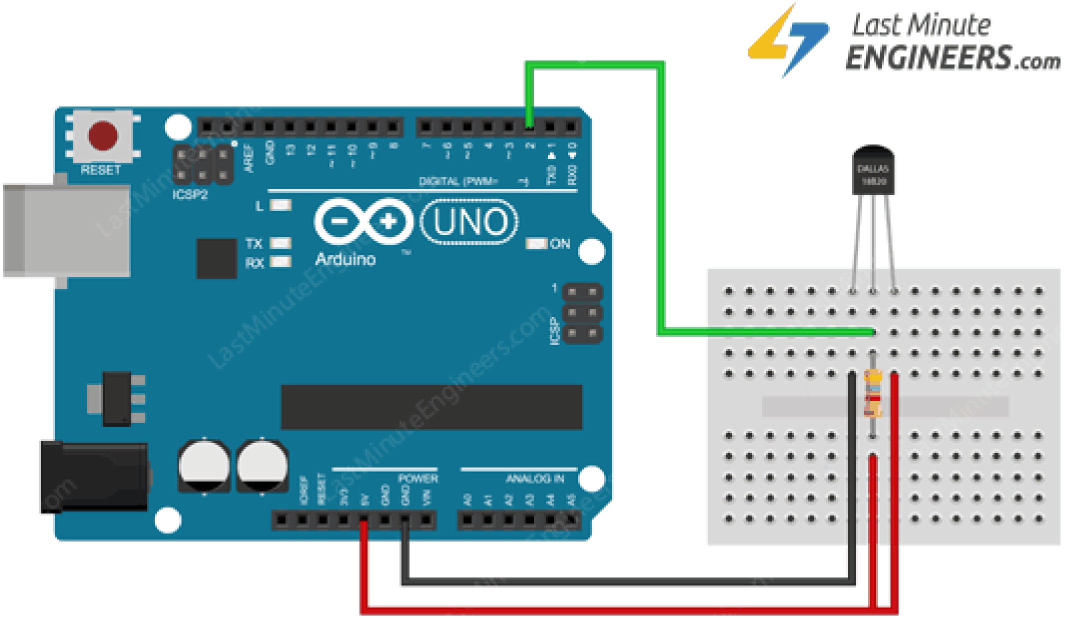
Wiring DS18B20 Temperature Sensor to Arduino

Enough of the theory, Let's Go Practical! Let's hook the DS18B20 up to the Arduino.

The connections are fairly simple. Start by connecting VDD to the 5V out pin on Arduino and GND to ground.

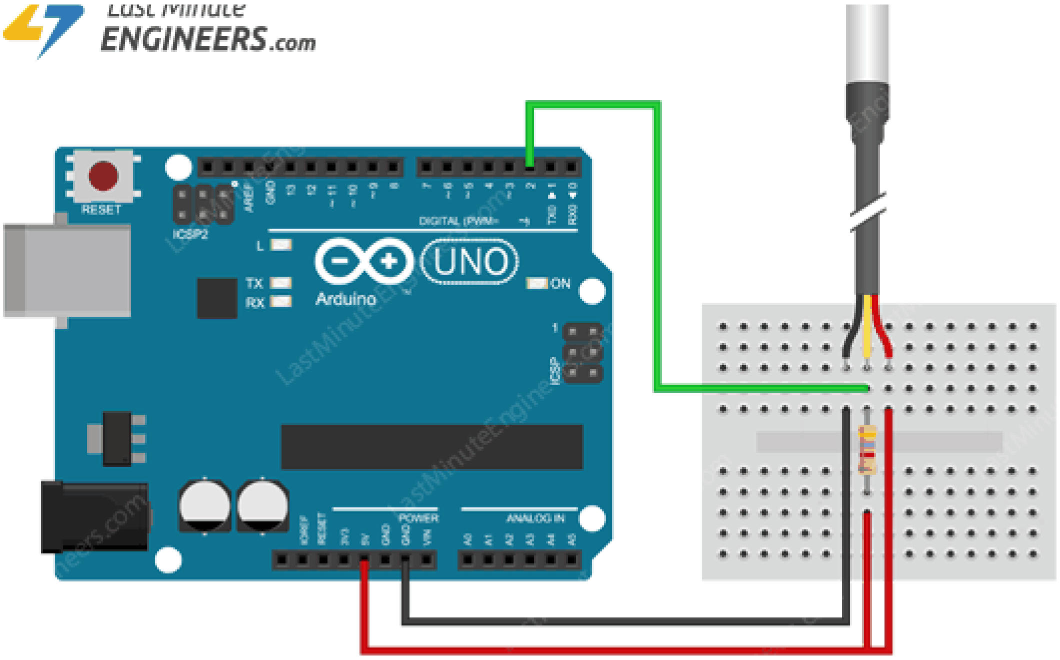
Next connect the remaining digital signal pin DQ to digital pin 2 on arduino. You'll also need to add the **4.7k pull-up resistor** between the signal and power pin to keep the data transfer stable. (internal pull-ups on the arduino does not work)

Be careful to get the DS18B20 the right way around. If you put it the wrong way around, it will get hot and then break.



Wiring DS18B20 Temperature Sensor to Arduino

If you are using the waterproof version of the DS18B20, connect Red stripe to 5V, Black connects to ground and Yellow Stripe is data that goes to digital pin 2 on arduino. You still need to connect a 4.7K pullup resistor from data to 5V.

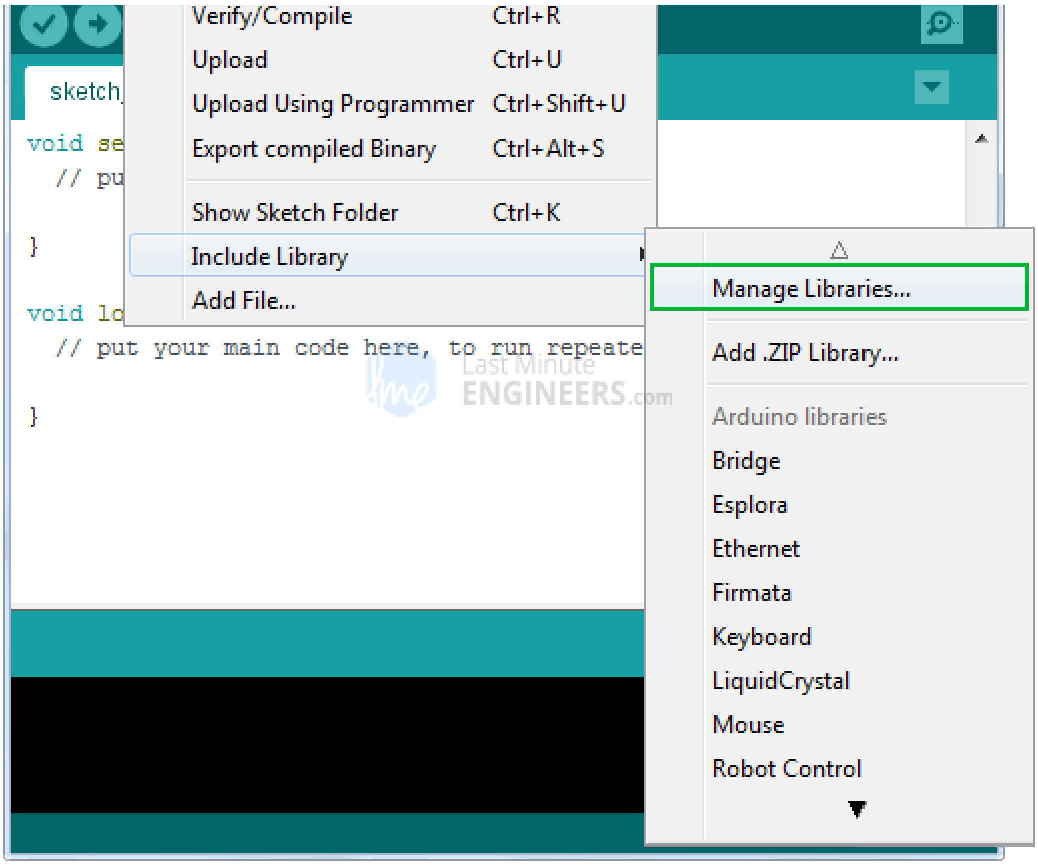


Wiring Waterproof DS18B20 Temperature Sensor to Arduino

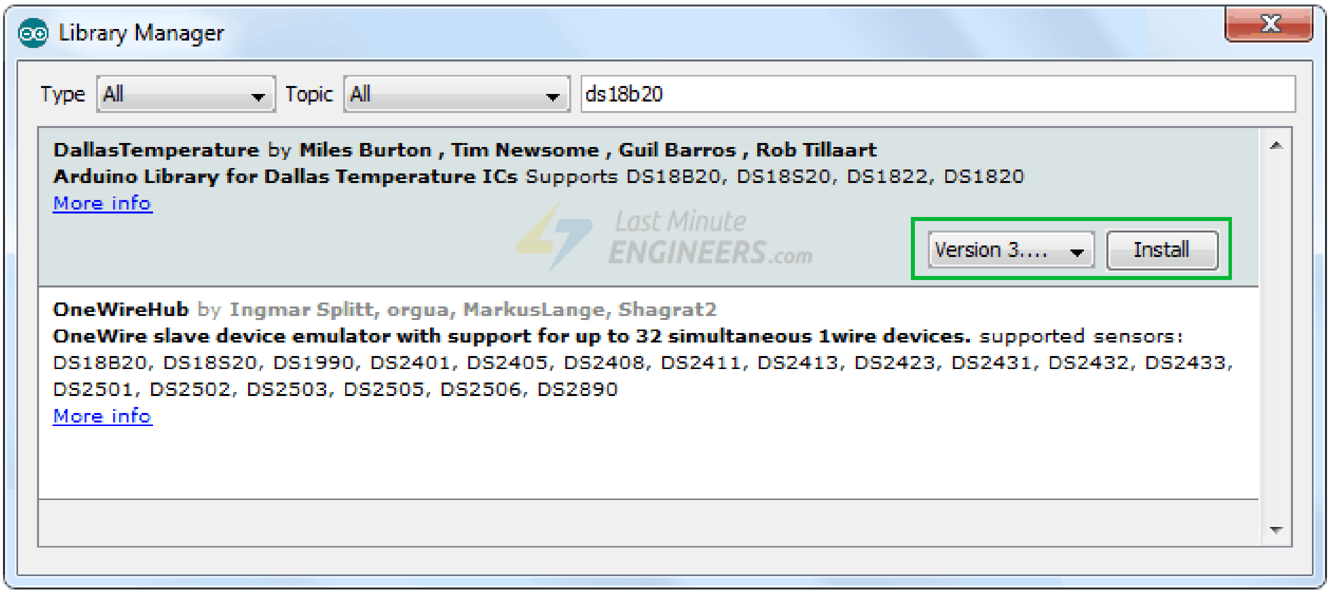
Installing Library For DS18B20

The Dallas 1-Wire protocol is somewhat complex, and requires a bunch of code to parse out the communication. To hide away this unnecessary complexity we will install DallasTemperature.h library so that we can issue simple commands to get temperature readings from the sensor.

To install the library navigate to the **Sketch > Include Library > Manage Libraries…**Wait for Library Manager to download libraries index and update list of installed libraries.



Filter your search by typing ‘**ds18b20**’. There should be a couple entries. Look for **DallasTemperature** by **Miles Burton**. Click on that entry, and then select Install.

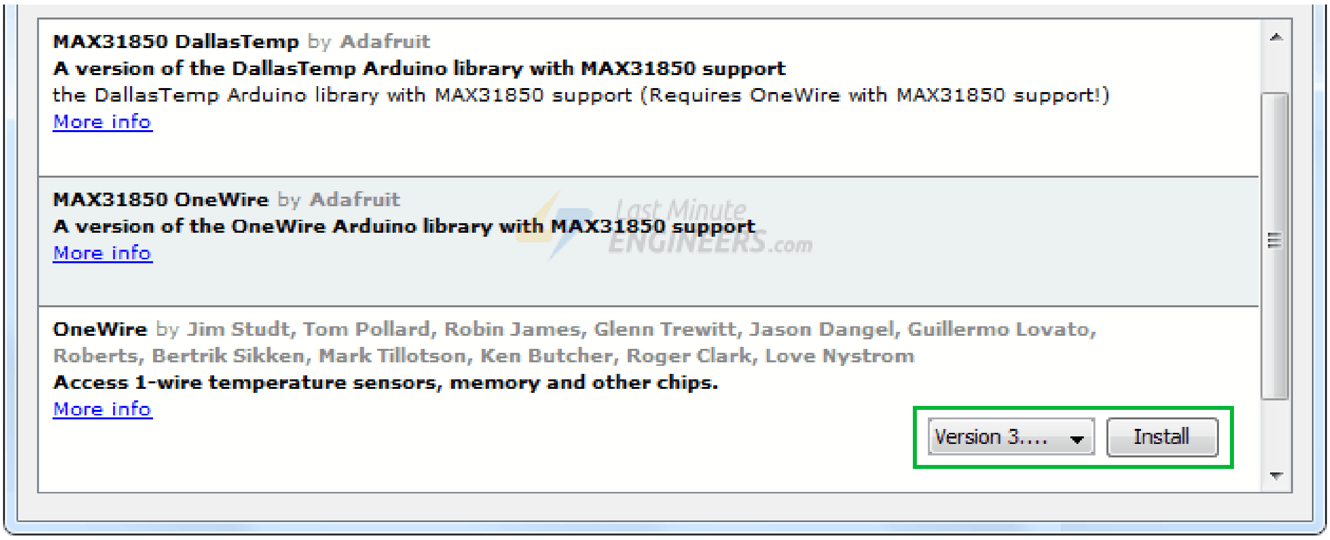


ds18b20

DallasTemperature by **Miles Burton , Tim Newsome , Guil Barros , Rob Tillaart**
Arduino Library for Dallas Temperature ICs Supports DS18B20, DS18S20, DS1822, DS1820
More info

OneWireHub by **Ingmar Splitt, orgua, MarkusLange, Shagrat2**
OneWire slave device emulator with support for up to 32 simultaneous 1wire devices. supported sensors: DS18B20, DS18S20, DS1990, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433, DS2501, DS2502, DS2503, DS2505, DS2506, DS2890
More info

This Dallas Temperature library is a hardware-specific library which handles lower-level functions. It needs to be paired with One Wire Library to communicate with any one-wire device not just DS18B20. Install this library as well.



MAX31850 DallasTemp by Adafruit
A version of the DallasTemp Arduino library with MAX31850 support
the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!)
More info

MAX31850 OneWire by Adafruit
A version of the OneWire Arduino library with MAX31850 support
More info

OneWire by Jim Studt, Tom Pollard, Robin James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Roberts, Bertrik Sikken, Mark Tillotson, Ken Butcher, Roger Clark, Love Nystrom
Access 1-wire temperature sensors, memory and other chips.
More info

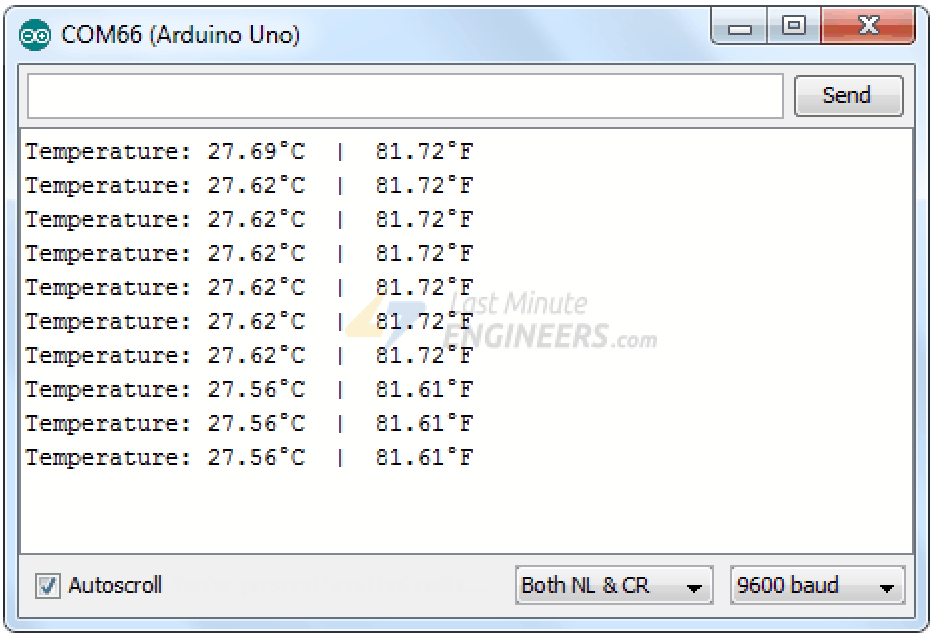
Arduino Code

The following sketch will give you complete understanding on how to read temperature readings from DS18B20 Temperature Sensor and can serve as the basis for more practical experiments and projects.

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into digital pin 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library
DallasTemperature sensors(&oneWire);

void setup(void)
{
  sensors.begin(); // Start up the library
  Serial.begin(9600);
}

void loop(void)
{
  // Send the command to get temperatures
```

```
    //print the temperature in Celsius
    Serial.print("Temperature: ");
```

Here’s how the output looks like in the serial monitor.



```
Temperature: 27.69°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.62°C  |  81.72°F
Temperature: 27.56°C  |  81.61°F
Temperature: 27.56°C  |  81.61°F
Temperature: 27.56°C  |  81.61°F
```

Code Explanation:

The sketch starts by including OneWire.h and DallasTemperature.h libraries and declaring the Arduino pin to which the sensor’s signal pin is connected.

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2
```

Next we create a one-wire object by passing sensor’s signal pin to its constructor. This one-wire object let’s us communicate with any one-wire device, not just DS18B20. In order to communicate with DS18B20 sensor, we need to create object of DallasTemperature library and pass reference of one-wire object as a parameter.

```
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

interact with the sensor.

- `begin()` function searches for connected sensors on the bus and sets bit resolution (12 bits) for each.
- `requestTemperatures()` function sends command for all sensors on the bus to perform a temperature conversion.
- `getTempCByIndex(deviceIndex)` function reads and returns temperature reading from the sensor. `deviceIndex` is nothing but the location of the sensor on the bus. If you are using only one DS18B20 on the bus, set it to 0.

Other useful functions in DallasTemperature.h library

There are a few useful functions you can use with DallasTemperature object. Few of them are listed below:

- `setResolution()` function sets resolution of internal Analog-To-Digital converter of DS18B20 to 9, 10, 11, or 12-bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively.
- `bool getWaitForConversion()` function returns the value of the waitForConversion flag. It can be useful when you want to check whether a temperature conversion is complete.
- `setHighAlarmTemp()` & `setLowAlarmTemp()` function sets the internal high & low temperature alarms for a device in degrees Celsius. Valid range is -55 to 125°C
- `bool hasAlarm()` function returns true if device have an alarm condition when temperature exceeds high & low alarm temperature setting.

Share

Disclaimer Privacy Policy