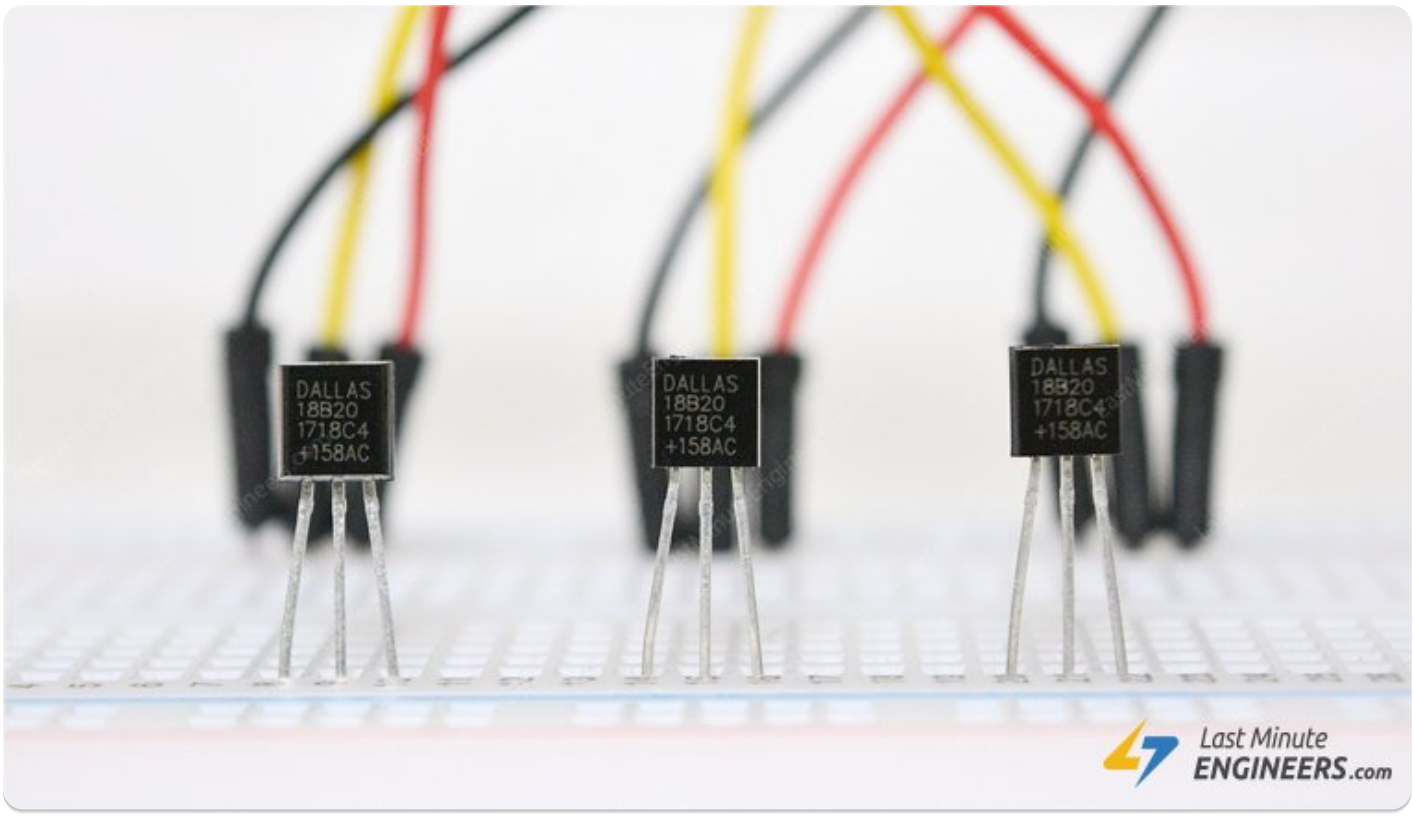
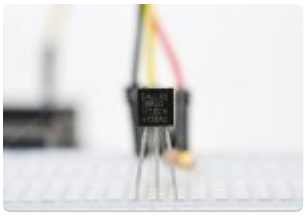
Interfacing Multiple DS18B20 Digital Temperature Sensors with Arduino



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 code** burned in at the factory, it's easier to differentiate them from one another.

The following tutorial demonstrates how to interface multiple DS18B20 on a single bus & get temperature readings from each of them. This feature can be a huge advantage when you want to control many DS18B20s distributed over a large area.

It may look intimidating, but you should be familiar with basics of DS18B20 one-wire temperature sensor, before venturing further into this tutorial. Consider reading through below tutorial first.



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...

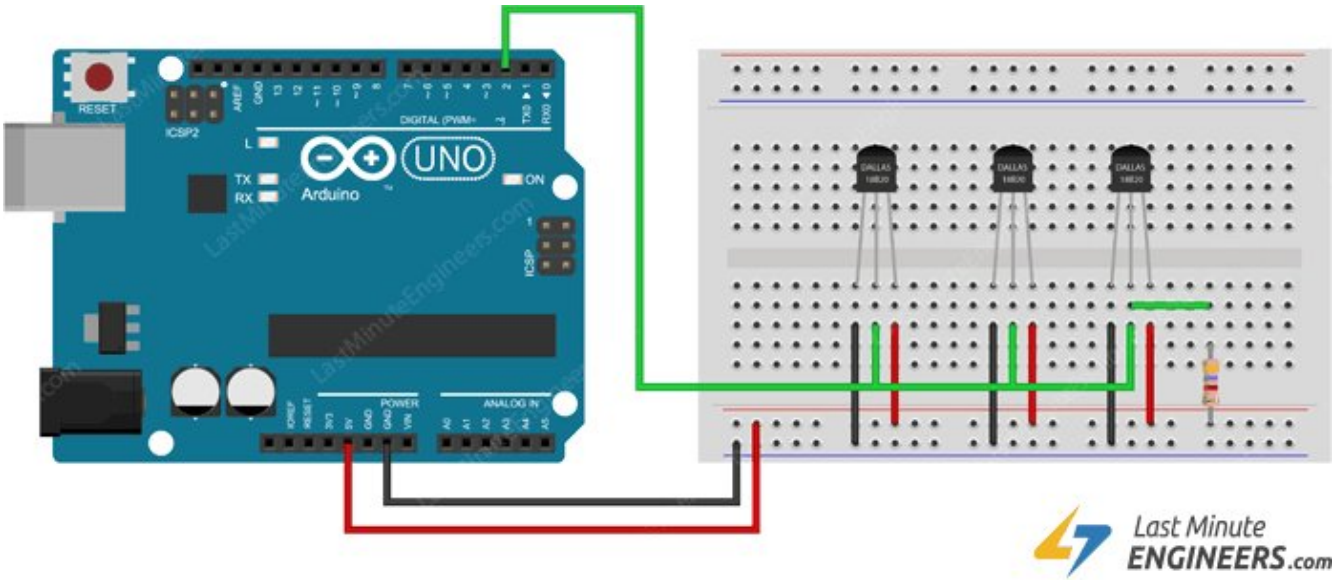
Without any further delay, let's hook DS18B20s to our Arduino.

Wiring Multiple DS18B20 Sensors to Arduino

Connections are fairly simple.

Start by connecting all the DS18B20s in parallel i.e. common all the VDD pins, GND pins & signal pins. Then connect VDD to the 5V out on Arduino, GND to Arduino ground and connect signal pin to digital pin 2 on arduino.

Next, you'll need to add one **4.7k pull-up resistor** for whole bus between the signal and power pin to keep the data transfer stable. (internal pull-ups on the arduino does not work)



Wiring Multiple DS18B20 Temperature Sensors With 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.

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.

Method 1: Reading DS18B20 By Index

In this method, when the Dallas Temperature library is initialized, it detects all the sensors sharing the same bus. It treats the whole bus as an array of sensors and assigns them an index. So that we can pinpoint each sensor by its index and read temperature.

```
#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);

int deviceCount = 0;
float tempC;

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

  // locate devices on the bus
  Serial.print("Locating devices...");
  Serial.print("Found ");
  deviceCount = sensors.getDeviceCount();
  Serial.print(deviceCount, DEC);
  Serial.println(" devices ");
```

The output of above sketch looks like

Code Explanation:

The sketch starts by including libraries, declaring pin to which sensor bus is connected and creating object of DallasTemperature library.

In setup part of code we first call `begin()` function. It initializes the bus and detects all the DS18B20s present on it. Each sensor is then assigned with an index and set bit resolution to 12-bit.

Next we call `getDeviceCount()` function to get the number of devices found on the bus.

In looping part of the code we use `requestTemperatures()` function to send command to all the sensors for temperature conversion.

Now using a simple `for(int i = 0; i < deviceCount; i++)` loop we can iterate through the array of sensors and read temperature of DS18B20 at an index `i` by simply calling `getTempCByIndex(i)`

Method 2: Reading DS18B20 By Address

We know that each DS18B20 has a unique 64-bit address assigned to it to differentiate them from one another. In this method, we'll **find that address** to label each sensor accordingly. The address can then be used to **read each sensor** individually.

Finding Addresses Of DS18B20s On Bus

The following sketch detects all the DS18B20s present on the bus and prints their one-wire address on the serial monitor.

You can wire just one sensor at a time to find its address (or successively add a new sensor) so that you're able to identify each one by its address. Then, you can label each sensor.

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

```
// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 2

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

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

// variable to hold device addresses
DeviceAddress Thermometer;

int deviceCount = 0;

void setup(void)
{
  // start serial port
  Serial.begin(9600);

  // Start up the library
  sensors.begin();

  // locate devices on the bus
```

Now, open the serial monitor. You should get something as follows

Copy all the addresses as we need them in out next sketch.

Reading DS18B20s By Address

The following sketch reads the temperature from DS18B20s by their addresses. Before you head for uploading the sketch, you need to change the addresses of DS18B20s with the one you've found in previous sketch.

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

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

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

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

// Addresses of 3 DS18B20s
uint8_t sensor1[8] = { 0x28, 0xEE, 0xD5, 0x64, 0x1A, 0x16, 0x02, 0xEC };
uint8_t sensor2[8] = { 0x28, 0x61, 0x64, 0x12, 0x3C, 0x7C, 0x2F, 0x27 };
uint8_t sensor3[8] = { 0x28, 0x61, 0x64, 0x12, 0x3F, 0xFD, 0x80, 0xC6 };

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
}

void loop(void)
{
  sensors.requestTemperatures();
```

The output of above sketch looks like

Code Explanation:

As usual the sketch starts by including libraries, declaring pin to which sensor bus is connected and creating object of DallasTemperature library.

Next, we enter the addresses that are found previously for each temperature sensor. In our case, we have the following.

```
uint8_t sensor1[8] = { 0x28, 0xEE, 0xD5, 0x64, 0x1A, 0x16, 0x02, 0xEC };
uint8_t sensor2[8] = { 0x28, 0x61, 0x64, 0x12, 0x3C, 0x7C, 0x2F, 0x27 };
uint8_t sensor3[8] = { 0x28, 0x61, 0x64, 0x12, 0x3F, 0xFD, 0x80, 0xC6 };
```

In setup section, we initialize the library by calling `begin()` function and initialize serial communication with PC.

In loop section, we simply send command to all the sensors for temperature conversion using `requestTemperatures()` function.

We then call `printTemperature(DeviceAddress deviceAddress)` custom function to print the temperature of the sensor whose `deviceAddress` is passed as parameter.

```
void printTemperature(DeviceAddress deviceAddress)
{
  float tempC = sensors.getTempC(deviceAddress);
  Serial.print(tempC);
  Serial.print((char)176);
  Serial.print("C  |  ");
  Serial.print(DallasTemperature::toFahrenheit(tempC));
  Serial.print((char)176);
```

```
  Serial.println("F");
}
```

The above function merely calls the library specific function `getTempC(deviceAddress)` to display temperature in Celsius and `DallasTemperature::toFahrenheit()` to display temperature in Fahrenheit.