Exploring One-wire Temperature sensor "DS18B20" with Microcontrollers

Author: Mohamed Fezari and Ali Al Dahoud
Badji Mokhtar Annaba University
University of Al-Zaytoonah Faculty of IT, Jordan

Abstract: in this article we will introduce a one wire temperature sensor, well suited for WSN apps and IoT Applications. DS18B20 is a temperature sensor which can measure temperature from -55°C to +125°C with an accuracy of +- 5%. It follows 1 wire protocol which has revolutionized the digital world. Because of it's 1 wire protocol, we can control multiple sensors from single pin of Microcontroller.

DS18B20 is generally used in industrial projects where high accuracy is necessary. We will present a detailed overview of this temperature sensor in today's post. Pin-out description, working mode, protocol etc will be explained. We will also share some links of projects where we have interfaced it with Arduino or other microcontrollers. Finally, some schematics applications are exposed.

DS18B20 Introduction

- DS18B20 is a digital temperature sensor which follows 1-wire protocol and can measure temperature from -55°C to +125°C (-67°F to +257°F) with an accuracy of +-5%.
- Data received from single wire is in the ranges of 9-bit to 12-bit.
- As DS18B20 follows 1-wire protocol so we can control this sensor via single pin of Microcontroller. (We also have to provide GND)
- 1-wire protocol is an advanced level protocol and each DS18B20 is equipped with a serial code of 64 bit which helps in controlling multiple sensors via single pin of microcontroller.
- In simple words, it assigns different addresses to all sensors attached and by calling the address, you can get that sensors' value.
- So, now let's have a look at DS18B20 Pin-out:

Pin Out description

- DS18B20 has 3 pins in total, which are:
  - Pin # 1: Vcc (We have to provide +5V here).
  - Pin # 2: Data Pin (It's the 1-wire from where we will get temperature readings).
  - Pin # 3: GND (We have to provide ground here).
- It is available in two packages, one is simple while the ot her one is waterproof DS18B20, both of their pinouts are shown in below figure:

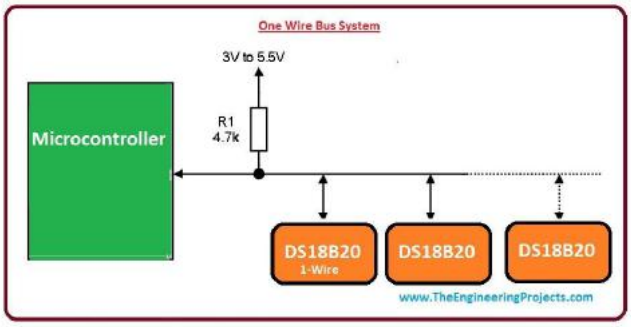
Now let's have a look at some of DS18B20 Characteristics & Features:

Features on DS18B20

Unique 1-Wire® interface requires only one port pin for communication .
Converts temperature to 12-bit digital word in 750ms (max.)
User-definable nonvolatile (NV) alarm settings

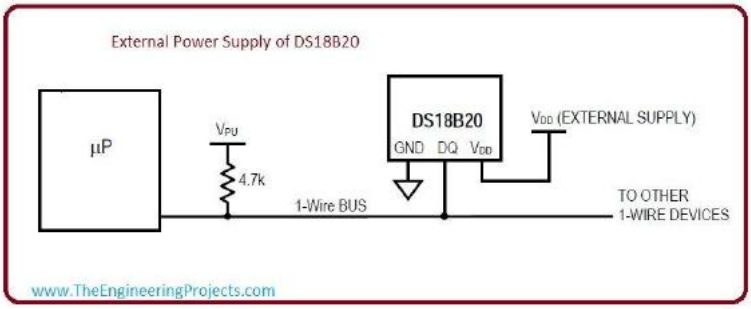
Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
Available in 8-pin SO (150mil), 8-pin µSOP, and 3-pin TO-92 packages
Software compatible with the DS1822
Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system One Wire Bus System

- Main advantage of 1 wire protocol is that we can control multiple 1-wire devices via single pin of Microcontroller.
- You must have heard of master slave system, where 1 master device can control or communicate with all slave devices.
- 1-wire protocol follows similar master slave system, where microcontroller acts as a master and all our 1-wire devices e.g. DS18B20 act as slaves.
- If we have interfaced only one device with our microcontroller then such a system is called **single drop** but if we interface multiple 1-wire devices via single pin then it's called **multi drop** system.
- Now lets have a better understanding of One Wire System from figure given below:



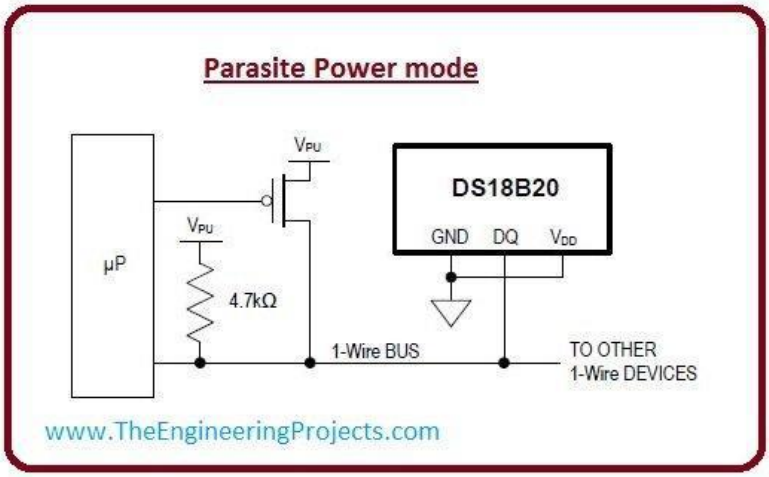
External Power Supply for DS18B20

- In this method, we provide power to DS18B20 by conventional method i.e. battery or adapter.
- This method is applicable for temperature below +100 degree Celsius.
- Main benefit of this method is, there is no extra load on resistor which use in this method and it perform work correctly.
- Lets, have a look at the connections in below figure:



Parasite Power Mode

- In this method we do not need special power supply.
- This method is used for temperature greater than +100 Celsius.
- In normal situation this method provide efficient current and voltage to DS18B20
- But, in special work when DS18B20 convert temperature value into digital then current value increase to such value which can damage resister.
- To limit current in save value and good working of DS18B20 it is necessary to use pull up Mosfet transistor.
- As, it is use only for specific temperature value there we use external power supply.



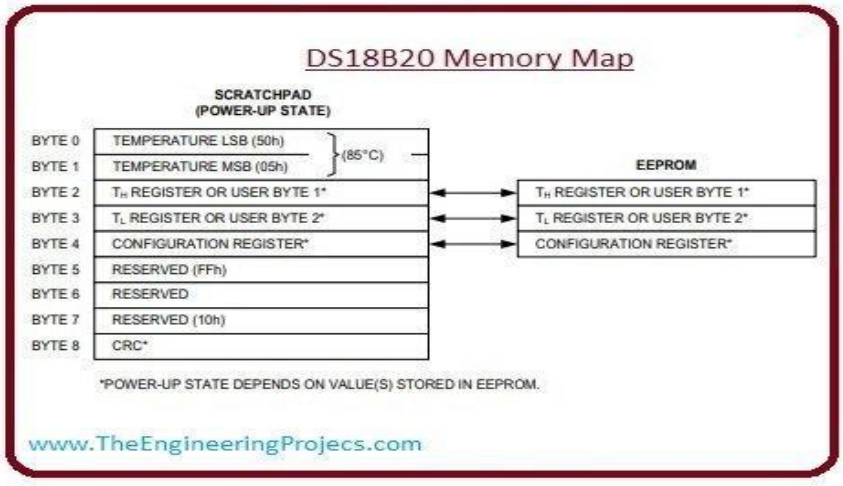
Now , lets have a look at picture of this method

How does DS18B20 Work?

- It works on the principle of direct conversion of temperature into digital value.
- Its main features is change its bit numbers according to change in temperature
- Like, it changes bit in 9. 10 ,11,and 12 bits as temperature changes in values 0.5 °C . 0.25°C,1.25 and 0.0625°C respectively.
- Its default bits value is 12 but it changes values according to temperature Change
- It has alarm and LCD as temperature changes alarms work and temperature value changes which we can get from lcd.
- Now lets have a look at DS18B20 memory map

Memory Map for the circuit

- There are two types of memories in DS18B20
- One SRAM and other is EEPROM.
- SRAM is volatile memory it has data only in on condition
- EEPROM is Non-volatile memory it stores data in off condition
- EEPROM also have low and high alarm trigger
- To have better understanding of Memory Map of DS18B20 we take a look at picture which give better idea of memory map of DS18B20 circuit.



Function Commands for the circuit

These are function Commands of DS18B20. These commands allow some to read and write data on DS18B20 scratched memory.

- **Convert T[44h] :** This command starts the single temperature conversion.

- **Write scratched Pad [4Eh]:** In this command we can write data on memory of DS18B20 to three bytes. Data is transferred in least multiple bit first.
- **Read Scratched Pad[BEh]:** In this command we can read data on scratched pad memory of DS18B20.
- **Copy Scratchpad [48h]:** This commands data from scratched pad and send data to EEPROM in 2 , 3 and 4 bytes.
- **Read Power Supply [B4h]:** This command tells about power supply mode of DS18B20.

Now lets have a look at Applications of DS18B20:

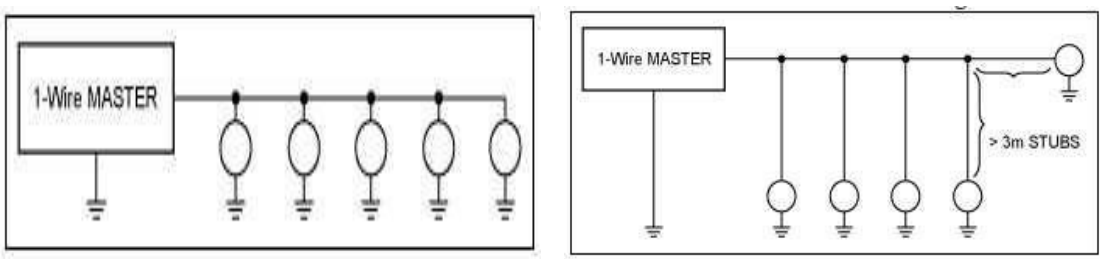
DS18B20 applications

DS18B20 is used for temperature measurement. There are some applications of DS18B20:

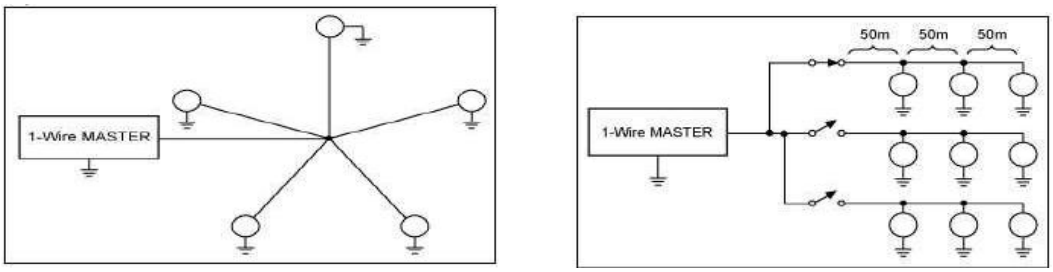
- It can be used in industries as temperature measuring device.
- It can be used as thermometer
- It can be used in thermostat controls system.
- We can use it in thermally sensitive devices.
- It can also use in HVAC systems.
- In network connections
- In embedded system for e-Health or wireless sensors network
- Best design of Temperature data logger

DS18B20 Network Connections

There are many different ways to connect up multiple devices which include the following layouts: Series, stubs, Star and switched.



Multiple Sensor Linear network with stubs



Multiple Sensor Star Network

Multiple Sensor Switched linear network

Other application and interfacing with microcontrollers:

With Arduino

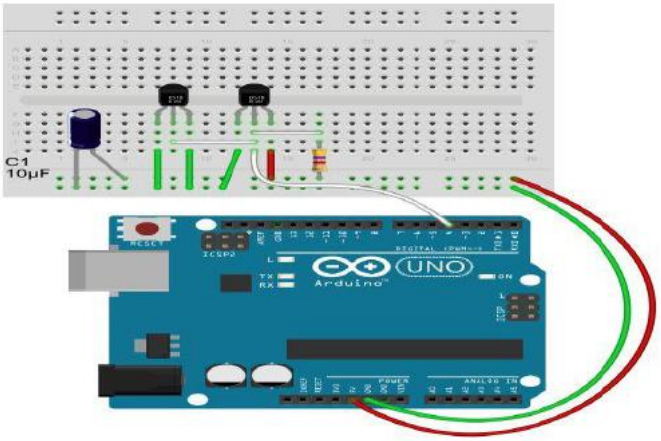
Circuit Layout: Arduino DS18B20 Multiple Device wire connection:

Both parasitic and external power connections can be used on the same One-wire bus.

The left hand device is parasitic powered (the so-called 1-wire interface which is actually a 2 - wire interface - GND and signal) while the right hand device is externally powered (3 - wire interface - GND, signal and power).

- You can remove either device from the solderless breadboard and you'll still get a temperature reading from the other one!

Arduino DS18B20 with multiple devices on a single bus wire:

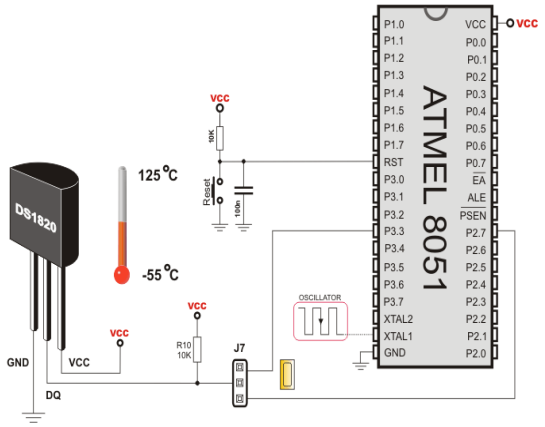


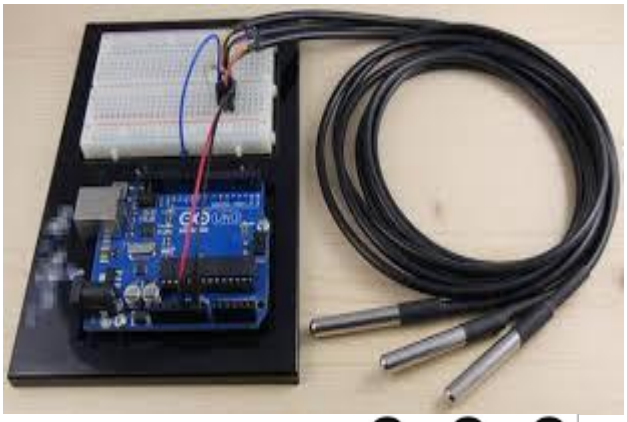
Code for Arduino Uno

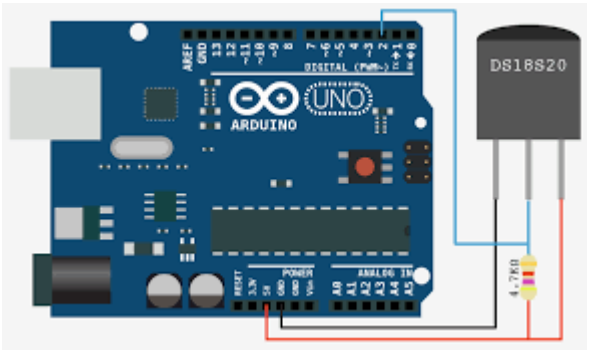
```
#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// http://www.pjrc.com/teensy/td libs OneWire.html
//
// The DallasTemperature library can do all this work for you!
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(4);  // on signal pin (a single 4.7K resistor is necessary)

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

void loop(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return;
  }
  Serial.println();

  // the first ROM byte indicates which chip
  switch (addr[0]) {
```

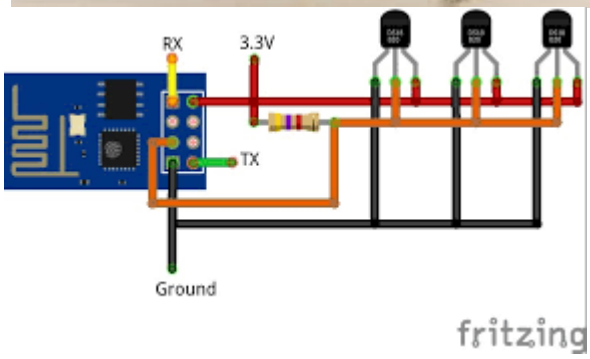
```
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);        // start conversion, with parasite power on at the end

  delay(1000);     // maybe 750ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of it.

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  Serial.print("  Data = ");
  Serial.print(present, HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print(OneWire::crc8(data, 8), HEX);
  Serial.println();

  // Convert the data to actual temperature
  // because the result is a 16 bit signed integer, it should
  // be stored to an "int16 t" type, which is always 16 bits
  // even when compiled on a 32 bit processor.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");
}
```

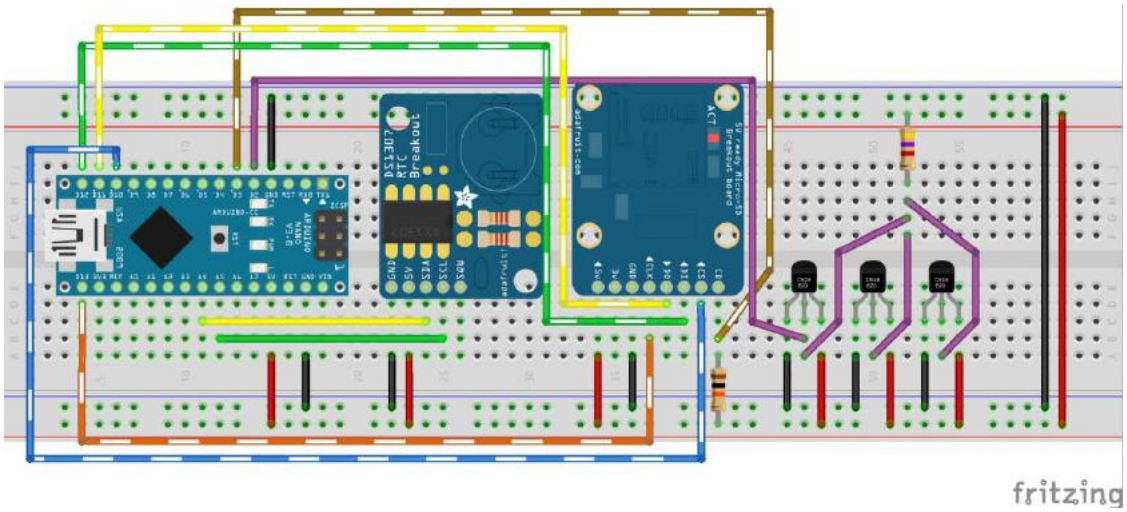
Some Images for Connection of DS18B20 with microcontrollers

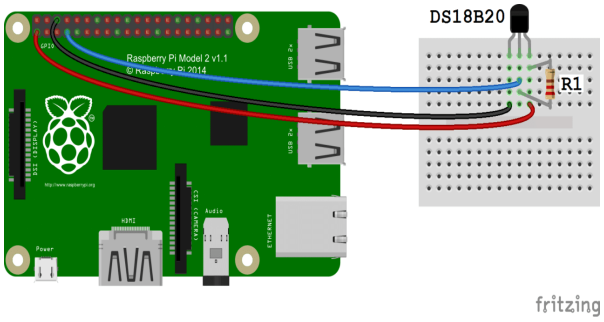


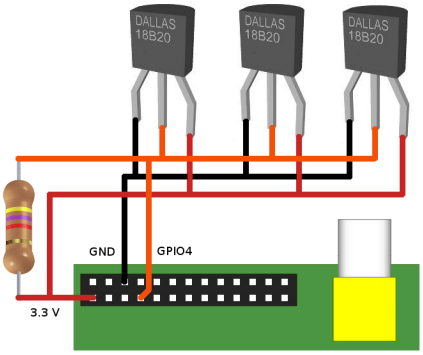












Conclusion : this small component is very well adapted to different design where we need to acquire temperature from -55 to 120 degrees, in this paper we presented the main information's we need to develop applications using this component, it well suited for applications using wireless sensors network and IoT. You can find also some references for more details on the circuit.

References:

1) https://www.codeproject.com/KB/system/1-wire_USB/DS18b20.pdf
2) https://www.best-microcontroller-projects.com/ds18b20.html
3) http://vascoferraz.com/projects/ds18b20-data-logger/
4) https://www.instructables.com/id/Arduino-Data-Logger-With-2-DS18B20-and-Sample-Rate/
5) https://www.cs.unb.ca/tech-reports/documents/TR15-235.pdf
6) https://www.best-microcontroller-projects.com/ds18b20.html
7) https://blog.bandinelli.net/index.php?post/2014/11/23/Temp%C3%A9rature-suivie-avec-un-Raspberry-Pi-B%2C-une-sonde-DS18B20-et-Munin
8) Vishesh Pamadi and Bradford G. Nickerson, "Getting Started With 1-Wire Bus Devices", TR15-235, University of New Brunswick Fredericton, N.B. E3B 5A3 Canada August 25, 2015