DS18B20 sensor de temperatura digital

Sensor de temperatura DS18B20

El sensor de temperatura DS18B20 es un dispositivo que se comunica de forma digital.
Cuenta con tres terminales, los dos de alimentación y el pin “data”.

Utiliza la comunicación OneWire el cual es del área de electrónica digital.

Básicamente se trata de un protocolo especial que permite enviar y recibir datos utilizando un solo cable a diferencia de la mayoría de los protocolos que requiere dos vías.

Sobre el sensor DS18B20…

- Es un termómetro digital de alta precisión entre 9 y 12 bits de temperatura en grados Celsius (el usuario puede escoger la precisión deseada).
- Su temperatura operativa se encuentra entre -50 y 125 grados Celsius.
La precisión, en el rango comprendido entre -10 y 85 grados es de ±0.5 grados.
- Su precio es económico su interfaz de funcionamiento es sencilla y su uso es muy provechoso para proyectos que requieran mediciones precisas y confiables.
- Para más información, consultar la hoja de datos(datasheet) del dispositivo:

http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

- Se puede escoger entre el modelo sumergible y los modelos para uso en placas de circuitos.

Con Arduino el código requiere de la utilización de 2 librerías que deben ser instaladas antes de cargar el código a la placa.
Estas son:

DallasTemperature
OneWire

Ejemplo básico simple.

```
//Se importan las librerías.
#include <OneWire.h>
#include <DallasTemperature.h>

//Se declara el pin donde se conectará la DATA.
#define Pin 2

//Se establece el pin declarado como bus para la comunicación OneWire.
OneWire ourWire(Pin);

//Se instancia la librería DallasTemperature.
DallasTemperature sensors(&ourWire);

void setup()
{

delay(1000);
Serial.begin(9600);
```

```
//Se inician los sensores.
sensors.begin();
}

void loop()

{

//Prepara el sensor para la lectura.
sensors.requestTemperatures();

//Se lee e imprime la temperatura en grados Celsius.
Serial.print(sensors.getTempCByIndex(0));
Serial.println(" grados Centigrados");

//Se lee e imprime la temperatura en grados Fahrenheit.
Serial.print(sensors.getTempFByIndex(0));
Serial.println(" grados Fahrenheit");

//Retraso de 1 segundo entre lecturas.
delay(1000);

}
```

Quizás no sea el sensor más preciso que hay en el mercado ni el más sencillo.
Sin embargo el sensor de temperatura DS18B20 tiene 3 características que lo hacen un componente deseable.

Su bajo precio entre 1€ y 3€.
La precisión de 0,5ºC en el rango de vida humana son factores importantes a la hora de elegir el DS18B20 para nuestros proyectos.
Permite la conexión en serie de bastantes sensores gracias al bus 1-Wire que viene implementado en el dispositivo.

Si a esto se añade que se puede comprar la versión que viene en forma de sonda impermeable, definitivamente se tiene un sensor muy versátil que puede llegar a medir temperaturas en entornos muy húmedos e incluso dentro de líquidos.

Sensor para líquidos con Arduino

Si bien se puede medir la temperatura con el LM35 el sensor de temperatura DS18B20 es uno de los sensores más versátiles que se puede encontrar en el mercado.
Este sensor es idóneo cuando se quiere medir la temperatura en ambientes húmedos e incluso dentro del agua.
Ya que hay una versión que viene en forma de sonda impermeable.
A lo largo de este artículo se ven las particularidades ventajas y desventajas del DS18B20.
No se trata de un sensor de temperatura común es algo más.

Quizás lo más complicado sea la programación ya que utiliza un protocolo poco común dentro del mundo de Arduino 1-Wire.
Pero gracias a las librerías la programación resulta muy sencilla.
El sensor se ha utilizado en el proyecto IoT Fridge Saver.

Como norma no solo se está aquí para copiar y pegar también aprender.
Tutorial del sensor de temperatura DS18B20.

Indice de contenidos

Características técnicas del sensor DS18B20

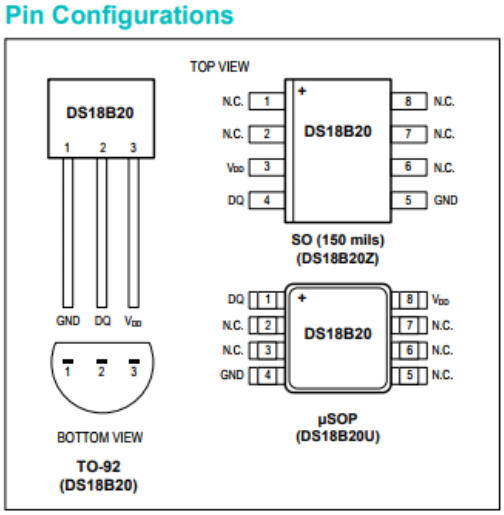
Dirigirse a la hoja de características técnicas del DS18B20.
En este documento todo lo necesario para conectar y programar el componente.

https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Hay diferentes formas del sensor DS18B20.
A esto se le llama encapsulado y depende de donde se utilice.
Protoboard, PCB, etc, elegir uno u otro.
Lo importante de esta parte es la disposición de los pines.
Para cada encapsulado hay una disposición diferente.
Existen 3 encapsulados y una sonda sumergible:

- TO-92
- SO
- µSOP
- Modelo sonda

El más adecuado para prototipar con Arduino es el TO-92 por su fácil conexión en la una protoboard.



sonda ds18b20

La única diferencia entre ellos es la forma o encapsulado.
Todos tienen 3 pines útiles VDD, GND y DQ.

- VDD

Es la tensión de alimentación.
Se puede alimentar desde 3V a 5,5V.
Este dato es mas que interesante cuando se quiere trabajar con placas como Arduino MKR1000 y NodeMCU que trabajan a 3,3V.

- GND

Es el negativo de la fuente de alimentación o toma de tierra.

- DQ
Es el pin de datos.
Por este pin es por donde se recibirán todos los datos en el protocolo 1-Wire.

Protocolo 1-wire

Este protocolo tiene una ventaja.
Como su propio nombre indica 1-Wire significa un cable en español solo es necesario utilizar un cable para conectar varios sensores de temperatura DS18B20.
Por lo tanto solo se utilizar 1 pin para conectar múltiples sensores y en Arduino también.

Conectar el sensor de temperatura DS18B20 a un Arduino

Es interesante conocer más aspectos técnicos de este sensor.
Es importante fijarse en el consumo eléctrico si se quiere utilizar el DS18B20 con alguna placa que requiera de baterías o pilas.
Para este caso no hace falta si es alimentado a través de la red eléctrica.

Rango de temperaturas del DS18B20

Es importante saber qué rango de temperaturas es capaz de medir cualquier sensor.
No es lo mismo medir la temperatura ambiente de una casa que medir la temperatura de un congelador o frigorífico.

También es importante conocer el error que puede llegar a tener y la resolución del sensor de temperatura DS18B20.
Toda esta información la sacamos de la hoja de características técnicas.
El DS18B20 puede medir temperaturas entre -55ºC y 125ºC.
Es un rango muy amplio sin embargo no en todo el rango tiene el mismo error.

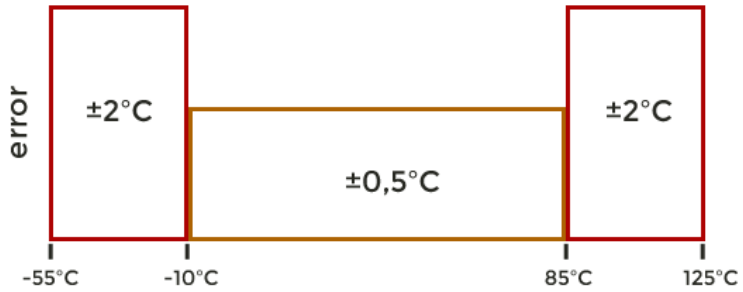
DS18B20 y el error en la medición

Un sensor de temperatura mediría exactamente la temperatura que hace en un sitio.
Todas las mediciones tiene error.
Por lo tanto todos los sensores no importa cual sea tienen un error de medición.

Un sensor de temperatura tiene errores debido a factores externos al ruido inherente en los circuitos eléctricos y alteraciones en el medio físico.

Aunque los componentes eléctricos tengan errores estos se pueden medir.
Por lo tanto sabemos más o menos cuanto oscilará la medición entorno a su valor real.
En el caso del DS18B20 de este tipo el error depende del rango de temperaturas.

Para temperaturas entre -10ºC y 85ºC se puede tener ±0,5ºC.
Para el resto de temperaturas entre -55ºC y 125ºC el error es de ±2ºC.



Error ds18b20

Esto equivale a decir que si el sensor DS18B20 suministra una temperatura de 23ºC el valor real estará entre 22,5ºC y 23,5ºC.
Si por el contrario suministra un valor de 90ºC el valor real estará entre 88ºC y 92ºC.

Esto tampoco debe preocupar a no ser que quiera controlar algún proceso industrial.

Resolución del sensor de temperatura DS18B20

Una de las características más interesantes de este sensor es que puede trabajar con diferentes resoluciones.
El concepto de resolución es cual es la variación mínima que se puede medir entre dos temperaturas.
Esto mismo que sucede con los pines analógicos en Arduino o cualquier otra placa de desarrollo.
El DS18B20 admite resoluciones de 9-bit, 10-bit, 11-bit y 12-bit.
Por defecto utiliza la resolución de 12-bit.

Las variaciones para cada resolución las puedes consultar en la siguiente tabla.

Resolución	Temperatura
9-bit	0,5ºC
10-bit	0,25ºC
11-bit	0,125ºC
12-bit	0,0625ºC

Elegir una resolución u otra dependerá de la precisión que se necesite para 1 proyecto.
A través de la programación se puede cambiar dicha resolución pero eso más adelante.

Características del sensor de temperatura DS18B20

* Identificador o dirección única para cada sensor.
Estos sensores poseen el identificador o dirección única para cada sensor.
Es una memoria de 64-bit equivalente a 8 bytes para almacenar este identificador único.

Esta dirección única es necesaria dentro del bus 1-Wire para identificar cada uno de los sensores de temperatura DS18B20 conectados al bus de comunicación.

El primer byte identifica el tipo de componente.
Por ejemplo para los DS18B20 es el número 28 en hexadecimal.

Con este tipo de comunicaciones se consiguen 2 cosas.

- Robustez en la transmisión de los datos ya que trabaja con datos digitales mucho menos sensibles a los efectos adversos del ruido que las señales analógicas.
- Permite conectar muchos sensores de temperatura con un único pin digital.

Internamente tiene otro tipo de memoria que sirve para diferentes cosas.

* Utiliza CRC.
Este se se almacena en la memoria.
Es el sistema de verificación de redundancia cíclica CRC para la detección de errores en los datos.

También almacena la temperatura obtenida.

* Dispone de 2 alarmas que se disparan si la temperatura es mayor o menor que un umbral de temperatura máxima o temperatura mínima.

Con todo estas características el DS18B20 se convierte en un sensor potente con unas capacidades superiores a otros en el mismo rango de precios.
El precio de este sensor oscila entre 1€ y 3€ casi despreciable.

Resumen de las características técnicas del DS18B20

Característica	Valor
Voltaje de alimentación	3V a 5,5V
VDD	voltaje de alimentación
GND	Tierra
DQ	Datos
Rango de temperaturas	-55ºC a 125ºC
Error rango -10ºC a 85ºC)	±0,5ºC
Error rango -55ºC a 125ºC)	±2ºC
Resolución programable	9-bit, 10-bit, 11-bit o 12-bit default

Conexión del DS18B20 con Arduino

Todo este montaje se hace entorno al protocolo 1-Wire.

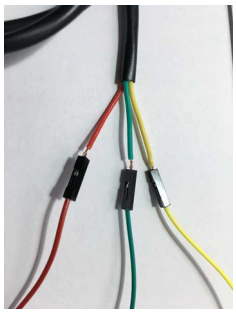
Esto no es único del DS18B20 cualquier otro componente que trabaje con el protocolo 1-Wire, utilizará la misma configuración eléctrica.

sonda ds18b20

En el TO-92 y más concreto el que viene en forma de sonda impermeable.
Si se tiene otro modelo o incluso el mismo sin sonda los pasos serán prácticamente los mismos.

Conectar los cables de la sonda DS18B20 a una protoboard o a Arduino

En este caso la sonda impermeable lleva unos cables normales sin conectores de metal.
Es la que mejor se adapta a la hora de prototipar con cualquier placa ya sea un Arduino o un ESP8266.



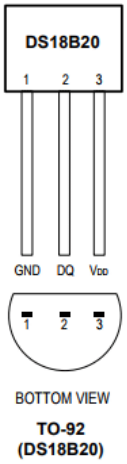
cables dupont ds18b20

Un cable dupont hembra este tipo de cables son los que suelen venir en casi todos los kit de Arduino y conectar cada terminal en cada cable dupont.

- Identificación de pines / cables del DS18B20

La distribución de los pines dependerá del encapsulado.
En cualquier caso siempre vamos a trabajar con 3 pines.
Un pin para alimentación VDD un pin para tierra GND y un pin para la transmisión de datos DQ.

Con el modelo TO-92 no hay problema.
Estos son los pines.
Sin embargo cuando se compra un DS18B20 en una sonda impermeable no se ve el propio sensor solo los cables.



pinout to-92

Para la sonda hay configuraciones de colores en los cables.
En concreto 3.

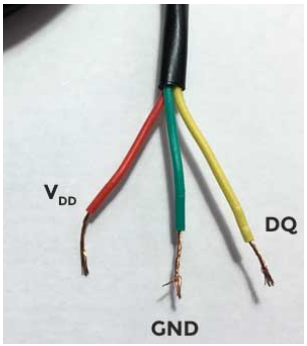
El tipo 1 donde:

- Cable rojo: VDD
- Cable negro: GND
- Cable amarillo: DQ

ds18b20 cables tipo a
Imagen obtenida del Datasheet Waterproof

El tipo 2 donde:

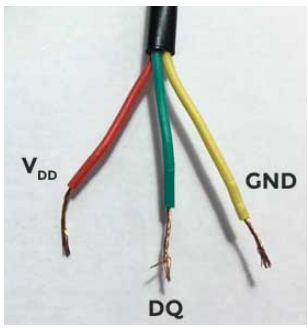
- Cable rojo: VDD
- Cable verde: GND
- Cable amarillo: DQ



ds18b20 cables tipo b

El tipo 3 donde:

Cable rojo: VDD
Cable verde: DQ
Cable amarillo: GND



ds18b20 cables tipo c

Si se tiene el tipo A no hay problema solo hay una configuración.
Si se tiene alguno de los otros hay un problema.

No se sabe como se debe de conectar.
La recomendación es primero buscar el fabricante tarea muy complicada si lo se ha comprado en Aliexpress o alguna otra tienda de China.
La segunda opción es probar las 2 conexiones para ver cual es la que funciona.

Elegir primero el tipo 2 es la configuración mas habitual.
Para finalizar esta sección un resumen de todas las conexiones.

Sensor tipo 1	Sensor tipo 2	Sensor tipo 3	Pin DS18B20
Negro	Verde	Amarillo	GND
Rojo	Rojo	Rojo	Vdd
Amarillo	Amarillo	Verde	DQ

Aunque los colores más típicos son los que he comentado aquí también se puede encontrar otras configuraciones de colores.
Hay sondas impermeables del DS18B20 donde el cable verde es de color azul.
Seguir los mismos pasos expuestos anteriormente para identificar cada conector.

Conexión sensor de temperatura DS18B20 con Arduino

Existen 2 formas de alimentar el DS18B20

- La primera sería la más simple y sería a través del pin VDD.
- La segunda opción sería hacerlo a través del propio pin de datos DQ en modo parásito.

En los 2 modos posibles siempre se pone una resistencia pull-up con el pin DQ.
El motivo de esta resistencia es debido a la electrónica para controlar el bus de comunicación.
Utiliza un FET de drenaje abierto que se comporta como una puerta AND.

Cuando todos los sensores conectados al bus 1-Wire no envíen datos la línea de datos será igual a la tensión que se suministre puede ser de 3V a 5,5V debido a la resistencia pull-up.

En el momento que un sensor empieza a transmitir la línea cambia de estado y hay un sensor transmitiendo datos.

Determinar la resistencia pull-up para tener en pin DQ siempre un modo alto o HIGH.
La resistencia que se vaya a utilizar dependerá de la longitud del cable.
Por norma general se utiliza siempre una de 4,7 kΩ.

En la siguiente tabla se puede ver la resistencia más adecuada dependiendo de la longitud del cable de datos.

Resistencia pull-up	Distancia del cable en metros
4,7 kΩ	0 m a 5 m
3,3 kΩ	5 m a 10 m
2,2 kΩ	10 m a 20 m
1,2 kΩ	20 m a 50 m

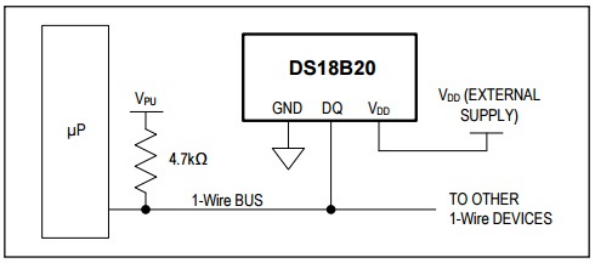
- Modo alimentación por el pin VDD

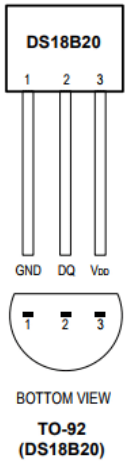
La alimentación por este pin es desde 3V a 5,5V.
No sobrepar el máximo valor o se dañael sensor de temperatura DS18B20.

El esquema de conexión para Arduino UNO sería el siguiente.

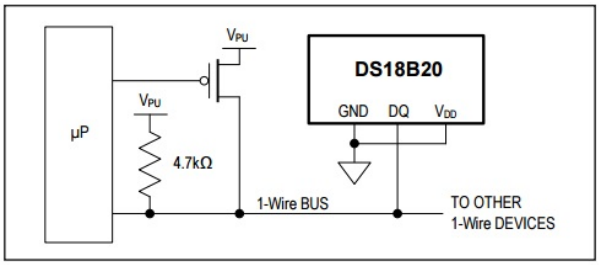
Circuito DS18B20 vdd

Con fuente externa:



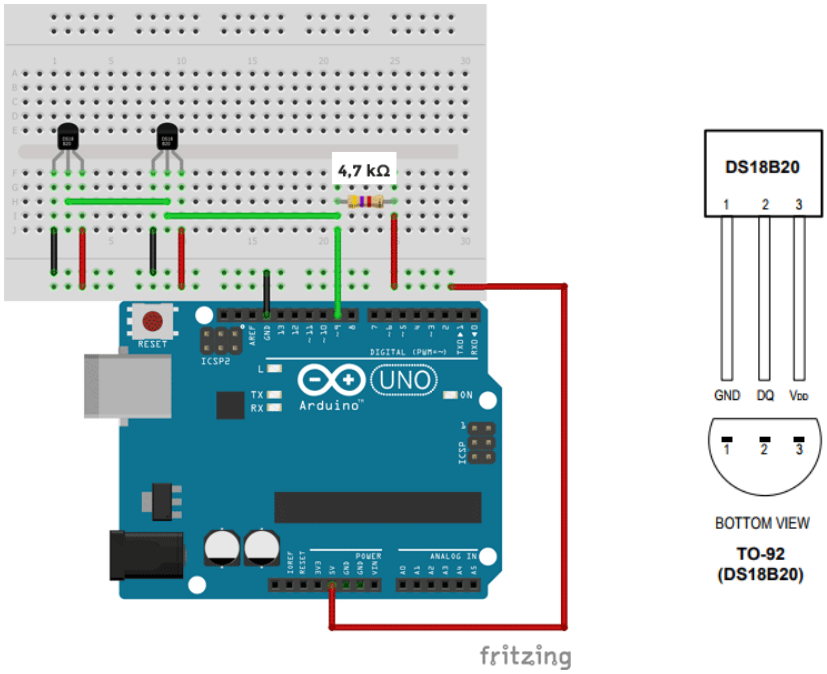


Modo parásito:

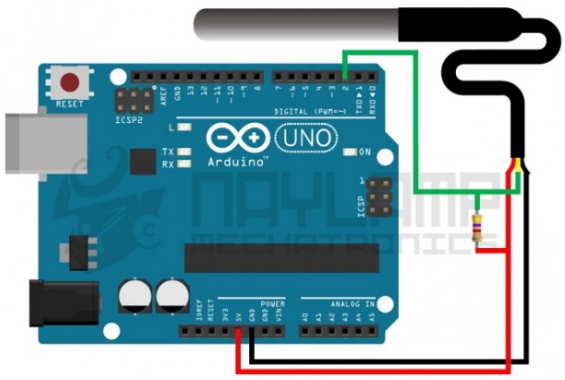


Circuito DS18B20 vdd fuente externa arduino uno

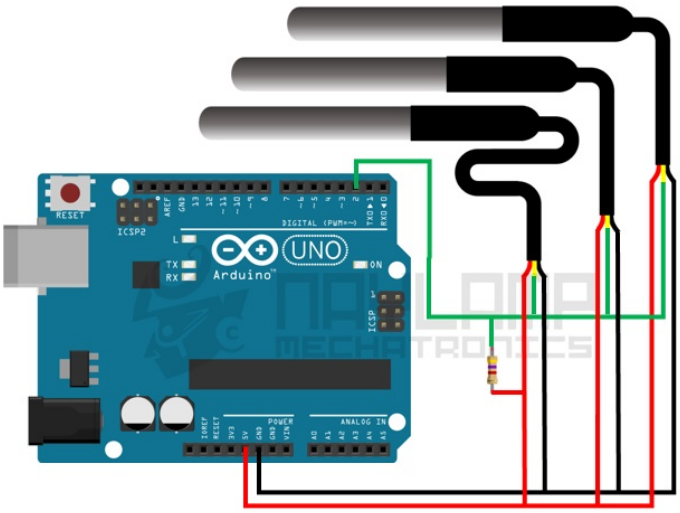
Con fuente externa.
En el circuito hay conectados 2 sensores DS18B20.
Uno de los pines va a tierra GND y el otra va a alimentación VDD.
El pin de datos DQ de todos los sensores al pin 9 del Arduino.



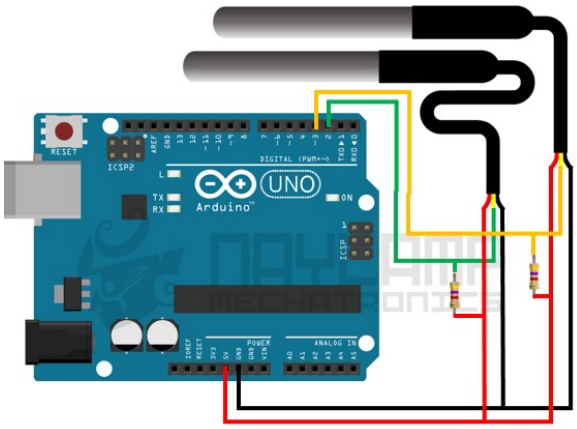
- Un solo sensor del tipo sumergible con fuente externa utilizando el pin 2 de datos del Arduino.



- Usa varios DS18B20 con fuente externa con un solo pin del Arduino.
En este caso se conectan todos los sensores al mismo bus 1-Wire.

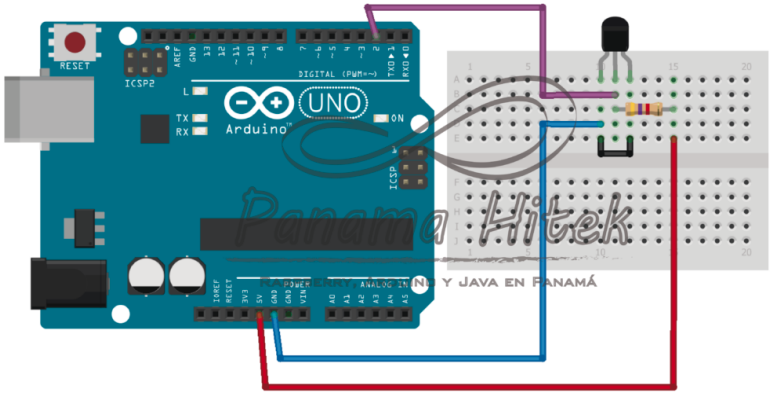


- Usa varios DS18B20 en diferentes pines del Arduino.
Cada sensor trabaja con un pin diferente y necesita su propia resistencia Pull-Up de 4.7K con fuente externa.



Modo alimentación modo parásito

A esta configuración se le llama `modo parásito`.
Es otra forma llamada modo parásito permite alimentar el sensor de temperatura DS18B20 a través del pin de datos DQ.
En este caso el pin de datos DQ de todos los sensores al pin 2 del Arduino.
El esquema de conexión para Arduino UNO sería el siguiente.



Circuito DS18B20 dq arduino uno

Observar los pines VCC y GND deben ir conectados entre sí.
La diferencia está en que el pin VDD se conecta a tierra.
Asegurarse de que todos los sensores tienen este pin a tierra GND.
La alimentación se introduce en el pin DATA por medio de una resistencia pull up que debe ser igual o mayor a 4.7 KOhm.

Con este último ejemplo el circuito preparado para empezar a programar.

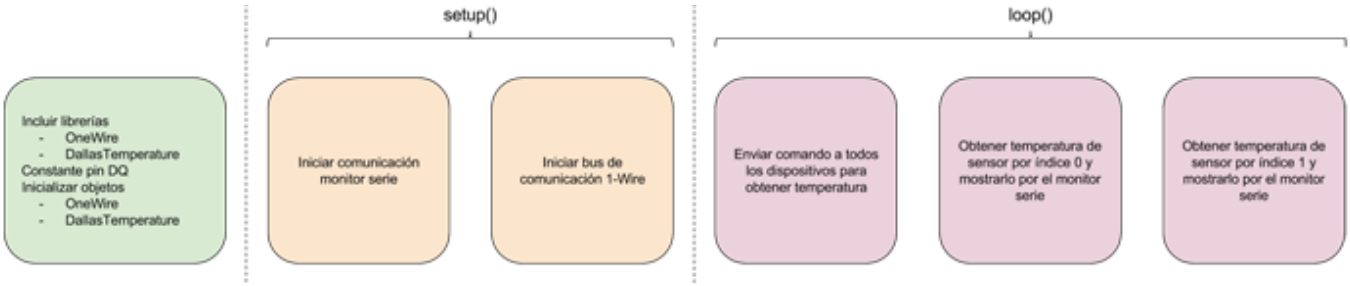
Da lo mismo el modo que se utilice en las 2 configuraciones de alimentación sea con fuente externa o parásita se programan igual.

Programando el sensor de temperatura DS18B20

Debido al protocolo 1-Wire que utiliza este sensor puede ser una tareas complejas del DS18B20.
Gracias a las librerías de Arduino resulta sencillo.

- Hacer un esquema de lo que se quiere hacer.
Esto siempre ayuda a programar.

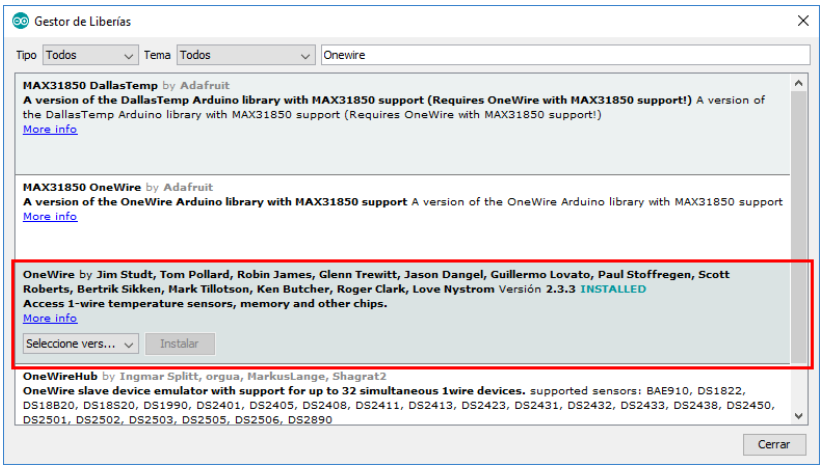
ESQUEMA PROGRAMACION DS18B20



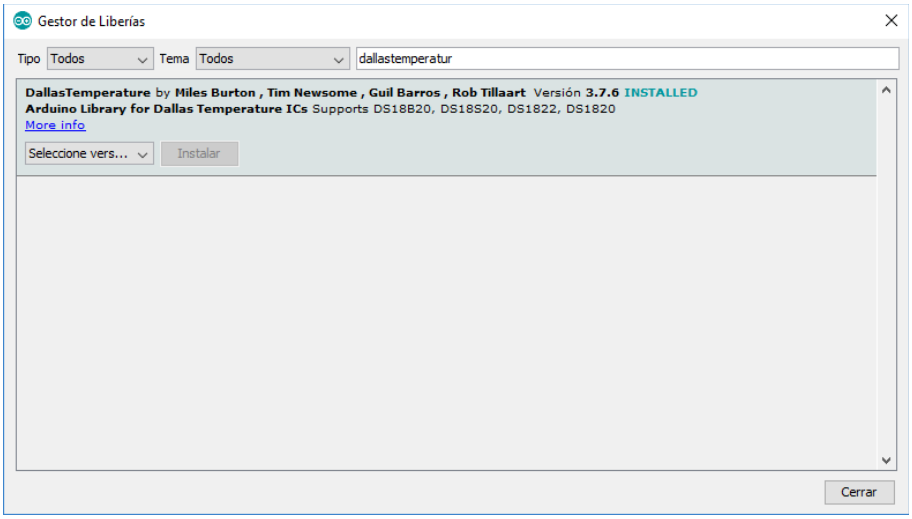
Librerías y variables del programa

La primera parte del programa hay que incluir las librerías OneWire y DallasTemperature. Buscar e instalar la librería a través del Gestor de Librerías.

Librería `onewire`.



Librería `dallastemperature`



El código de esta parte el siguiente.

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

```
#include <DallasTemperature.h>

// Pin donde se conecta el bus 1-Wire
const int pinDatosDQ = 9;

// Instancia a las clases OneWire y DallasTemperature
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);
```

Primero se incluyen las librerías en el código.

```
OneWire
DallasTemperature
```

OneWire es la que implementa el protocolo 1-Wire.
DallasTemperature implementa el código para enviar los comandos adecuados a los sensores y obtener la temperatura.

Por norma siempre se utiliza una variable se aconseja que sea una constante para indicar el número del pin donde se conectan los sensores.
Este pin es el marcado como DQ.

Se declaran dos objetos de las clases OneWire y DallasTemperature.
Solo se utiliza este último que es el que da acceso a los sensores de temperatura DS18B20.

Cuando una variable se pasa con el símbolo &nombreVariable se está pasando la posición de memoria donde se almacenan los datos y no su valor.
A esto se le llama paso por referencia.
Esto es lo que se hace al pasar el argumento al constructor de la clase DallasTemperature.

Función setup()

El código de la función setup() es sencillo.

```
void setup()

{
    // Iniciar la comunicación serie.
    Serial.begin(9600);
    // Iniciar el bus 1-Wire.
    sensorDS18B20.begin();
}
```

Se inicia la comunicación con el monitor serie indicando la velocidad de transmisión con Serial.begin().
Luego se inicia el bus de comunicación 1-Wire llamando a la función que no admite ningún parámetro.

```
sensorDS18B20.begin()
```

Función loop()

En esta función accede a los sensores a través del bus 1-Wire.

Muestra los datos por el monitor serie.

```
void loop()

{
    // Comandos de lectura de temperatura a los sensores.
    Serial.println("Comandos de lectura");
    sensorDS18B20.requestTemperatures();

   // Lee y muestra los datos de los sensores DS18B20.
    Serial.print("Temperatura sensor 0: ");
    Serial.print(sensorDS18B20.getTempCByIndex(0));
    Serial.println(" C");
    Serial.print("Temperatura sensor 1: ");
    Serial.print(sensorDS18B20.getTempCByIndex(1));
    Serial.println(" C");

    delay(1000);
}
```

Se muestra un mensaje para indicar que se envían comandos a los sensores para leer la temperatura. Para esto se llama a la función sensorDS18B20.requestTemperatures().

Los sensores han tomado la temperatura y la tienen lista para enviar a Arduino.
Recordar que los sensores estos almacenan este valor en una memoria interna.
Para solicitar dicha temperatura se hace una llamada a la función:

```
sensorDS18B20.getTempCByIndex(indice_sensor)
```

Analizar esta función.
Si se tienen varios sensores conectados con Arduino se los puede identificar de diferentes maneras.
La más rápida es a través de un índice.
A cada uno de ellos se le asigna un número en orden secuencial.

Si se tienen 2 sensores sus índices serán 0 y 1.
Si se tuvieran 5 sensores sus índices serían 0, 1, 2, 3 y 4.
Así sucesivamente.
Si tenemos solo se tiene un sensor el índice a utilizar es el 0.

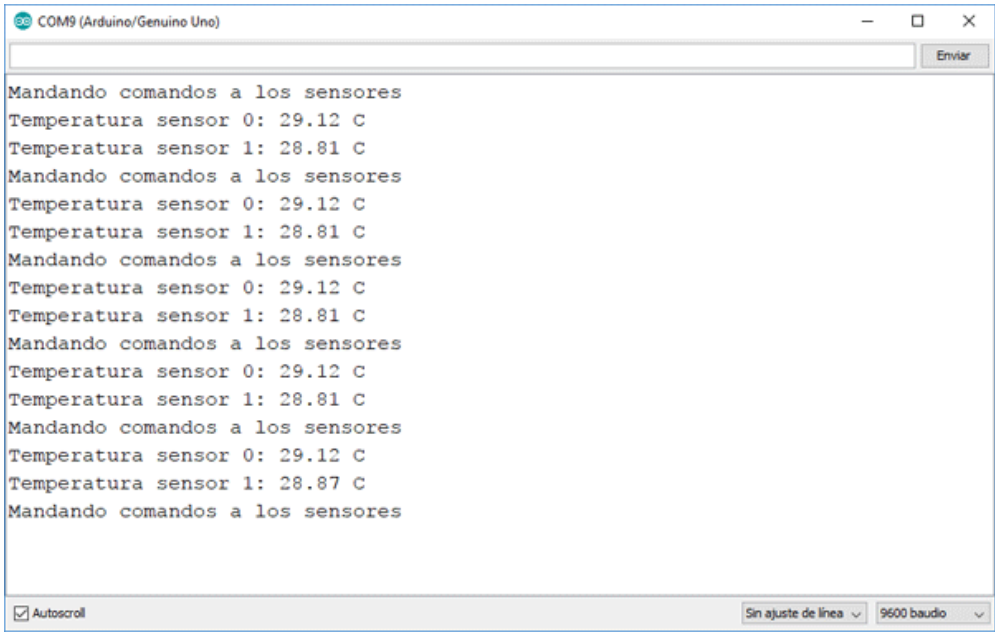
Además esta función devuelve la temperatura en grados Celsius.
Esto viene indicado por TempC.

- Código completo

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

// Pin bus 1-Wire.
const int pinDatosDQ = 2;

// Instancia a las clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);
```

```
void setup() {
    // Iniciar comunicación serie.
    Serial.begin(9600);
    // Iniciar el bus 1-Wire.
    sensorDS18B20.begin();
}

void loop() {
    // Comandos lectura de temperatura a los sensores.
    Serial.println("Comandos de lectura");
    sensorDS18B20.requestTemperatures();


   // Leer y mostrar los datos de los sensores DS18B20.
    Serial.print("Temperatura sensor 0: ");
    Serial.print(sensorDS18B20.getTempCByIndex(0));
    Serial.println(" C");
    Serial.print("Temperatura sensor 1: ");
    Serial.print(sensorDS18B20.getTempCByIndex(1));
    Serial.println(" C");

    delay(1000);
}
```

Cargando este código debe aparecer algo parecido a esto en el monitor serie.



salida temperatura ds18b20

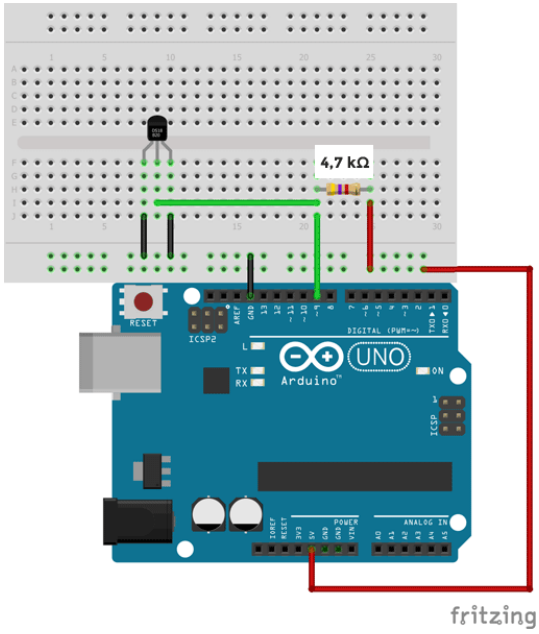
Implementación práctica

Imaginar que se quiere monitorear un frigorífico tanto la parte de arriba como el congelador.
Te gustaría saber cual es cada sensor.
Cada uno de los sensores está identificado con un número de 64-bit el identificador o dirección única.

Lo ideal sería identificar cada uno de los sensores que hemos conectado a Arduino eso es lo que se hace a continuación.

Identificar cada sensor de temperatura DS18B20

Para identificar un único sensor partir del siguiente esquema eléctrico donde solo se tiene un sensor conectado.
Esto se tiene que repetir con cada sensor.
Se utiliza la configuración de alimentación modo parásito alimentación por el pin de datos DQ.



Circuito DS18B20 dq arduino uno identificación

Se podría optar por otras opciones como diferenciar los sensores a través de la temperatura.
Por ejemplo poner uno en una estufa y el otro lo pones en temperatura ambiente.
Se verá como se notan las diferencias.
Sin embargo a nivel de código esto puede resultar más complicado.

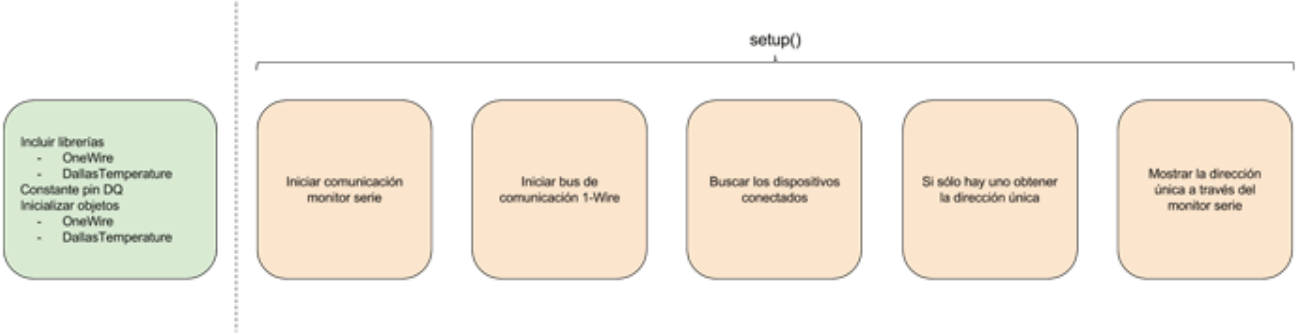
Lo importante de esta sección es identificar la dirección única de 64-bit de 8 bytes de cada sensor.
Debido a que la implementación de la librería DallasTemperature es más rápida cuando se trabaja con estos identificadores y no con los índices que asigna de forma automática.

Programar la identificación del DS18B20

Se muestra un diagrama de lo que se va a programar con este circuito.
Normalmente no es necesario hacer esto sobre todo si ya se tiene experiencia programando Arduino.

Pero si se está empezando es más que recomendable hacer un esquema de lo que se tiene que hacer antes de empezar.
En este caso lo siguiente.

ESQUEMA IDENTIFICACIÓN DS18B20



Librerías y variables del programa

La primera parte es la misma que en el ejemplo anterior.
Incluye las `librerías` `OneWire` y `DallasTemperature` y `declara` las `variables` y `clases`.

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

// Pin del bus 1-Wire.
const int pinDatosDQ = 2;

// Instancia a las clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);
```

Función setup()

En este caso se hace todo en la función setup().
Solo se quiere saber la dirección única que tiene el sensor conectado a Arduino.

```
void setup() {
    // Iniciar monitor serie y sensor de temperatura DS18B20.
    Serial.begin(9600);
    sensorDS18B20.begin();

    // Buscar sensores conectados.
    Serial.println("Buscando dispositivos...");
    Serial.print("Encontrados: ");
    int numeroSensoresConectados = sensorDS18B20.getDeviceCount();
    Serial.print(numeroSensoresConectados);
    Serial.println(" sensores");

   // Si se encuentra uno mostrar su dirección.
    if(numeroSensoresConectados==1){

        // Se define Tipo como un array de 8 bytes (uint8_t).
        DeviceAddress sensorTemperatura;
        // Obtener dirección.
        sensorDS18B20.getAddress(sensorTemperatura, 1);
```

```
      // Mostrar en el monitor serie.
         Serial.print("Sensor encontrado: ");

         // Recorrer los 8 bytes del identificador único.
         for (uint8_t i = 0; i < 8; i++)
         {
           // Si solo tiene un dígito rellenar con un cero a la izquierda.
           if (sensorTemperatura[i] < 16) Serial.print("0");

           // Mostrar los datos que van en HEXADECIMAL.
           Serial.print(sensorTemperatura[i], HEX);
         }
    }
}
```

Se inicializa el `monitor serie` y el `bus` de `comunicación 1-Wire` con `Serial.begin()` y `sensorDS18B20.begin()`.

- Comprobar si hay algún `sensor` conectado al `bus 1-Wire`.
- Hacer esto con el método `sensorDS18B20.getDeviceCount()`.
Este método devuelve un número entero con los sensores conectados al bus.

Puede ser muy útil para saber que todo está funcionando correctamente.
Imagínar que se tiene 3 sensores conectados a Arduino.
Lo primero que hace el programa es comprobar que estén conectados si no lo estuvieran se podría parar la ejecución y alertar de que algo está mal.

Lo siguiente es comprobar que solo se tenga conectado un único sensor a través de la sentencia de control if(…).

Recordar que este programa solo sirve para detectar la dirección de un único sensor.

Una vez comprobado se declara una variable del tipo DeviceAddress que no es más que un array de 8 bytes.
La dirección única tiene 64-bit 8 bytes y la almacena en esta variable.
Cada byte en un elemento del array.

La propia librería ya incorpora un tipo de dato específico para esta dirección DeviceAddress.
Es conveniente utilizarlas así no se tendrás ningún error y quedará todo claro.

Todo esto se muestra a través del monitor serie.
Debido a que es un array de 8 elementos se recorre con un bucle for(…).
Luego se muestra cada dígito en el monitor serie teniendo en cuenta de que si un número es menor que 16 se rellena con un cero a la izquierda.

Esto es debido a que en el sistema hexadecimal los primeros 16 números solo tienen 1 dígito:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Ejecutando el código de identificador del DS18B20

Código completo de la aplicación.

```
//Programa identificacion id de sensores DS18B20

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

// Pin del bus 1-Wire.
const int pinDatosDQ = 2;

// Instancia a las clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);

void setup()

{

    // Iniciar monitor serie y sensor de temperatura DS18B20.
    Serial.begin(9600);
    sensorDS18B20.begin();
}

void loop()

{

    // Buscar cantidad de sensores conectados.
    Serial.println(" ");
    Serial.println(" ");
    Serial.println("Buscando dispositivos...");
    Serial.println("Encontrados: ");
    int numeroSensoresConectados = sensorDS18B20.getDeviceCount();
    Serial.print(numeroSensoresConectados);
    Serial.println(" sensor / sensores");


  // Si ha encontrado uno mostrar su dirección.
    if(numeroSensoresConectados==1){

    // Tipo definido como una array de 8 bytes (uint8_t).
        DeviceAddress sensorTemperatura;
        // Obtener dirección
        sensorDS18B20.getAddress(sensorTemperatura, 1);

     // Mostrar por el monitor serie.
        Serial.print("Sensor encontrado: ");

    // Recorrer los 8 bytes del identificador único.
        for (uint8_t i = 0; i < 8; i++)
        {
          // Si solo tiene un dígito rellenar con un cero a la izquierda.
          if (sensorTemperatura[i] < 16) Serial.print("0");

          // Mostrar los datos que van en HEXADECIMAL.
          Serial.print(sensorTemperatura[i], HEX);
          Serial.print(" ");

        }
    }
```

```
}
```

Si se carga el código anterior en tu Arduino deberá aparecer algo parecido a esto en el monitor serie.

```
Buscando dispositivos...
Encontrados: 1 sensores
Sensor encontrado: 28A8F8E708000091
```

O este otro tipo de salida según se acomoden los print

```
Buscando dispositivos...
Encontrados:
1 sensor / sensores
Sensor encontrado: 284DBF300500007C
```

La dirección única será diferente para cada sensor de temperatura DS18B20.
Esa es la información importante que se tendrá que guardar y relacionar con el sensor.
Se puede pegar con una etiqueta o como quieras hacerlo.
Luego en el código se utiliza para obtener la temperatura de ese sensor.

identificacion ds18b20

Como se ve en la foto se puede poner algo descriptivo que indique donde se va poner ese sensor.

Una vez identificado el sensor ya se puede conectar otro e ir identificando cada uno de ellos.
En este caso hay 4 a utilizar para el proyecto IoT Fridge Saver.
2 sondas van al frigorífico y al congelador.
Otras 2 toman la temperatura de la parte delantera y trasera del electrodoméstico.

multiples ds18b20

Por último ver como acceder a los datos de temperatura a través de la dirección única de cada sensor de temperatura DS18B20.

Acceder a la temperatura del DS18B20 por Id del sensor
Por dirección única

Antes en el ejemplo se accede a la temperatura del DS18B20 a través del índice que ocupa en el bus de comuncicación.
Esto es un poco peligroso ya que no se tiene control sobre ese orden podría variar con alguna modificación en el circuito.

Por eso es importante conocer la dirección única.
En la sección anterior ya hemos identificado cada sensor programar y ser conscientes de la temperatura que da cada uno de ellos.

Librerías y variables del programa

Si el identificador obtenido es: `284DBF300500007C`

Reescribirlo de la siguiente manera:

```
{0x28, 0x4D, 0xBF, 0x30, 0x05, 0x00, 0x00, 0x7C}
```

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

// Pin del bus 1-Wire.
const int pinDatosDQ = 2;

// Instancia a las clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);

// Variables con las direcciones únicas de los 4 sensores DS18B20.
```

```
DeviceAddress sensorFrigorifico = {0x28, 0x4D, 0xBF, 0x30, 0x05, 0x00, 0x00, 0x7C};
DeviceAddress sensorCongelador = {0x28, 0x6F, 0xB6, 0xC6, 0x08, 0x00, 0x00, 0x3F};
DeviceAddress sensorDelante = {0x28, 0xA4, 0x29, 0xE6, 0x08, 0x00, 0x00, 0xF0};
DeviceAddress sensorAtras = {0x28, 0x45, 0x92, 0xE6, 0x08, 0x00, 0x00, 0xD1};
```

Las librerías ya se conocen OneWire y DallasTemperature.
Se tiene que declarar una constante para el pin DQ.
Se realizan las instancias de las 2 clases OneWire y DallasTemperature.

Como son 4 sensores en este ejemplo en la parte final se declaran 4 variables del tipo DeviceAddress.
Este tipo representa un array de 8 elementos.
Cada uno de ellos es 1 byte.
Ahora rescatar las direcciones únicas que se obtuvieron anteriormente.

Para asignar el valor a cada variable se tienen que hacer grupos de 2.
Cada grupo irá en un elemento del array y en total son 8 elementos que equivalen a 8 bytes 64-bit.

Fíjarse bien en los nombres aquí se pone de manifiesto la utilidad de las variables.
Asignar a cada dirección un descriptivo.
Esta variable almacenará la dirección única pero será fácilmente recordada gracias la nombre de la variable.

Nombrar variables es algo personal.
La recomendación es que acostumbrarse a poner nombres lo más descriptivo posible.

Función setup()

```
void setup()

{
  // Iniciar comunicación serie.
  Serial.begin(9600);
  // Iniciar bus 1-Wire.
  sensorDS18B20.begin();


 // Buscar la cantidad de sensores conectados.
  Serial.println("Buscando dispositivos...");
  Serial.println("Encontrados: ");
  Serial.print(sensorDS18B20.getDeviceCount());
  Serial.println(" sensores");
}
```

Comenzar la función setup() iniciando el monitor serie con Serial.begin() y el protocolo 1-Wire con sensorDS18B20().

Como se ha visto en el ejemplo de identificación se busca el valor de todos los dispositivos conectados al bus con la función sensorDS18B20.getDeviceCount().
En este caso debería dar 4 sensores si fueran menos es que hay algún error de conexión.
En las pruebas realizada se pueden tener problemas con las conexiones.
En el momento que falla alguno de los sensores puede ocurrir que tire abajo el bus 1-Wire y se desconectan todos los sensores.

Función loop()

```
void loop()

{
  // Comandos medición temperatura.
  Serial.println("Comandos de lectura");
  sensorDS18B20.requestTemperatures();

 // Leer y mostrar datos de los sensores DS18B20 por dirección única.
  Serial.print("Temperatura sensor frigorifico: ");
  Serial.print(sensorDS18B20.getTempC(sensorFrigorifico));
  Serial.println(" C");
  Serial.print("Temperatura sensor congelador: ");
  Serial.print(sensorDS18B20.getTempC(sensorCongelador));
  Serial.println(" C");
  Serial.print("Temperatura sensor delante: ");
  Serial.print(sensorDS18B20.getTempC(sensorDelante));
  Serial.println(" C");
  Serial.print("Temperatura sensor detras: ");
  Serial.print(sensorDS18B20.getTempC(sensorAtras));
  Serial.println(" C");

  delay(1000);
}
```

En la función `loop()` accede a la temperatura de los 4 sensores `DS18B20`.
Comienza enviando comandos pertinentes para que los sensores empiecen a medir.
Esto se hace con la función `sensorDS18B20.requestTemperatures()`.

Mostrar la temperatura de cada sensor.
Como ya se tiene identificado cada uno de ellos muestra dónde es esa temperatura.
Para acceder a la temperatura ya no utilizamos la función getTempCByIndex().
Utiliza `sensorDS18B20.getTempC()` al que se le pasa como `argumento` el `array` de `8 bytes` definido por el `tipo DeviceAddress`.

Se hace una `pausa` de 1 `segundo` para cada `medición`.
Si se quiere `almacenar` la `temperatura` en una `variable` debe ser del `tipo float`.

```
float temperaturaCongelador = sensorDS18B20.getTempC(sensorCongelador);
```

Para cambiar la resolución del sensor a: 9, 10, 11 o 12 bits. solo se debe usar la función:

```
sensors.setResolution(Address, 9); // resolución de 9 bits
```

Código completo

En este ejemplo es solo para 4 sensores.
Sustituir por el id que corresponda según el caso.

```
// Programa para obtener temperatura por id del sensor DS18B20.

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

```
// Pin del bus 1-Wire.
const int pinDatosDQ = 2;

// Instancia clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);

// Variables con las Id direcciones únicas de 4 sensores DS18B20.
DeviceAddress sensorFrigorifico = {0x28, 0x4D, 0xBF, 0x30, 0x05, 0x00, 0x00, 0x7C};
DeviceAddress sensorCongelador = {0x28, 0x6F, 0xB6, 0xC6, 0x08, 0x00, 0x00, 0x3F};
DeviceAddress sensorDelante = {0x28, 0xA4, 0x29, 0xE6, 0x08, 0x00, 0x00, 0xF0};
DeviceAddress sensorAtras = {0x28, 0x45, 0x92, 0xE6, 0x08, 0x00, 0x00, 0xD1};

void setup()

{
  // Iniciar comunicación serie.
  Serial.begin(9600);
  // Iniciar bus 1-Wire.
  sensorDS18B20.begin();


 // Buscar la cantidad de sensores conectados.
  Serial.println("Buscando dispositivos...");
  Serial.println("Encontrados: ");
  Serial.print(sensorDS18B20.getDeviceCount());
  Serial.println(" sensores");
}

void loop()

{
  // Comandos medición temperatura.
  Serial.println("Comandos de lectura");
  sensorDS18B20.requestTemperatures();


 // Leer y mostrar datos de los sensores DS18B20 por dirección única.
  Serial.print("Temperatura sensor frigorifico: ");
  Serial.print(sensorDS18B20.getTempC(sensorFrigorifico));
  Serial.println(" C");
  Serial.print("Temperatura sensor congelador: ");
  Serial.print(sensorDS18B20.getTempC(sensorCongelador));
  Serial.println(" C");
  Serial.print("Temperatura sensor delante: ");
  Serial.print(sensorDS18B20.getTempC(sensorDelante));
  Serial.println(" C");
  Serial.print("Temperatura sensor detras: ");
  Serial.print(sensorDS18B20.getTempC(sensorAtras));
  Serial.println(" C");

  delay(1000);
}
```

La salida es algo así:

```
Comandos de lectura
Temperatura sensor frigorifico: 23.81 C
Temperatura sensor congelador: -127.00 C
```

```
Temperatura sensor delante: -127.00 C
Temperatura sensor detras: -127.00 C
```

Observar que la lectura de de los sensores con valores `-127.00 C` de este ejemplo indica o que no están o están funcionando mal.

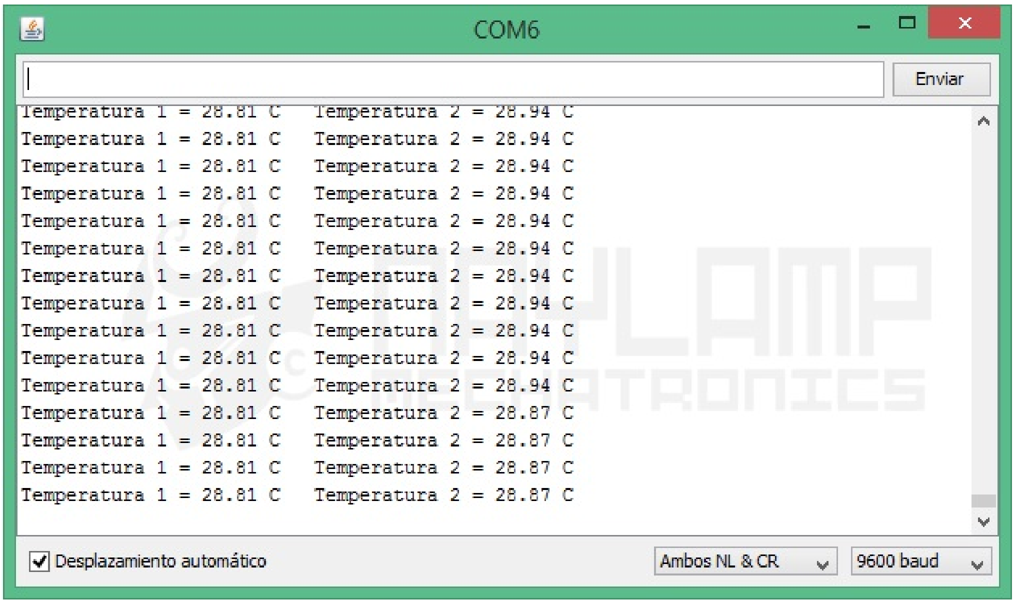
Otros ejemplos

Usando varios DS18B20 en diferentes pines del Arduino.

Cada sensor trabaja con un pin diferente y necesita su propia resistencia Pull-Up de 4.7K.
El código para realizar las lecturas es el siguiente:

```
// Programa con 2 sensores DS18B20 en distintos pines one-wire.
# include <OneWire.h>
# include <DallasTemperature.h>

// Se establece el pin 2 y el 3 como bus OneWire.
OneWire ourWire1(2);
OneWire ourWire2(3);

// Se declaran una variable u objeto para cada sensor el 1 y el 2.
DallasTemperature sensors1(&ourWire1);
DallasTemperature sensors2(&ourWire2);

void setup()

{

// Se inicia el monitor serie.
Serial.begin(9600);
// Se inicia el sensor 1 y 2.
sensors1.begin();
sensors2.begin();
// Se establece un tiempo entre lecturas.
delay(1000);

}

void loop()

{

// Comando de lectura de temperatura.
sensors1.requestTemperatures();
// Se obtiene la temperatura en ºC del sensor 1.
float temp1= sensors1.getTempCByIndex(0);

// Idem sensor 2.
sensors2.requestTemperatures();
float temp2= sensors2.getTempCByIndex(0);

Serial.print("Temperatura 1 = ");
Serial.print(temp1);
Serial.print(" C");
Serial.print("   Temperatura 2 = ");
Serial.print(temp2);
Serial.println(" C");
delay(100);
```

```
}
```



Varios DS18B20 con un solo pin del Arduino

En este caso se conectan todos los sensores al mismo bus 1-Wire.
averiguar la dirección de cada sensor para poder identificarlo.
El siguiente sketch solo se utiliza para obtener la dirección de los dispositivos conectados en el bus 1-wire.

```
// Programa para identificar Id sensores DS18B20.
#include <OneWire.h>

//Se establece el pin 2  como bus OneWire
OneWire ourWire(2);

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

void loop(void) {
  byte addr[8];
  Serial.println("Obteniendo direcciones:");
  while (ourWire.search(addr))
  {
  Serial.print("Address = ");
  for( int i = 0; i < 8; i++) {
    Serial.print(" 0x");
    Serial.print(addr[i], HEX);
  }
  Serial.println();
}

Serial.println();
```

```
ourWire.reset_search();
delay(2000);
}
```

Funciones importantes de librería DallasTemperature

Se enumeran y explican las funciones más importantes de la librería DallasTemperature.
Hay muchas más que se pueden inspeccionar en el código de la librería en GitHub.

```
uint8_t getDeviceCount(void)
```

Devuelve el número de dispositivos conectados al bus 1-Wire en un entero sin signo de 8-bit.

```
// Busca sensores conectados
Serial.println("Buscando dispositivos...");
Serial.println("Encontrados: ");
Serial.print(sensorDS18B20.getDeviceCount());
Serial.println(" sensores");

bool getAddress(uint8_t*, uint8_t)
```

Obtiene la dirección única de un dispositivo dado un índice en el bus 1-Wire.
Devuelve verdadero (true) si se encuentra el dispositivo.

// Tipo definido como una array de 8 bytes (uint8_t)

```
DeviceAddress sensorTemperatura;

// Obtener dirección
sensorDS18B20.getAddress(sensorTemperatura, 0);

bool isConnected(const uint8_t*)
```

Comprueba si un sensor está conectado al bus 1-Wire dado una dirección única.

```
// Tipo definido como una array de 8 bytes (uint8_t)
DeviceAddress sensorTemperatura = {0x28, 0xA8, 0xF8, 0xE7, 0x08, 0x00, 0x00, 0x91};
// Obtener temperatura por la dirección del sensor
if(sensorDS18B20.isConnected(sensorTemperatura)){
  // Obtener temperatura
  float temperatura = sensorDS18B20.getTempC(sensorTemperatura);
}

uint8_t getResolution()
```

Obtiene la resolución global de todos los sensores.
Devuelve un valor entero sin signo de 8-bit con la resolución.
Esta puede ser de 9-bit, 10-bit, 11-bit o 12-bit.

```
// Obtener la resolución global.
uint8_t resolucionGlobal =  sensorDS18B20.getResolution();
```

```
void setResolution(uint8_t)
```

Establece la resolución de un sensor dado su índice en el bus 1-Wire.
No devuelve ningún valor.

```
// Establece la resolución del sensor con índice 0.
sensorDS18B20.setResolution(0);
```

```
uint8_t getResolution(const uint8_t*)
```

Obtiene la resolución de un sensor dado su dirección única.
Devuelve el valor de la resolución.

```
// Obtenemos la resolución global
uint8_t resolucionGlobal =  sensorDS18B20.getResolution(sensorCongelador);
```

Hay 2 funciones con el mismo nombre getResolution().
A esto se le llama sobrecargar un método y la clase que lo implementa es lo suficientemente lista como para saber cual tiene que ejecutar dependiendo del número de argumentos que se le pase a la función.

```
void requestTemperatures(void)
```

Envía los comandos a los sensores para que hagan la lectura de la temperatura.

```
// Comandos medición temperatura a los sensores.
Serial.println("Mandando comandos a los sensores");
sensorDS18B20.requestTemperatures();
```

```
bool requestTemperaturesByAddress(const uint8_t*)
```

Igual que la función anterior pero este solo envía los comandos para que un solo dispositivo haga la lectura de temperatura dado una dirección única.

```
// Mandamos comandos para toma de temperatura del sensor del congelador
Serial.println("Mandando comandos a los sensores");
sensorDS18B20.requestTemperaturesByAddress(sensorCongelador);
```

```
bool requestTemperaturesByIndex(uint8_t)
```

Igual que los 2 anteriores pero dado el índice del sensor de temperatura DS18B20 dentro del bus 1-Wire.

```
// Mandamos comandos para toma de temperatura del sensor 0
Serial.println("Mandando comandos a los sensores");
sensorDS18B20.requestTemperaturesByIndex(0);

Serial.println("Mandando comandos a los sensores");
sensorDS18B20.requestTemperaturesByIndex(0);
```

```
float getTempC(const uint8_t*)
```

Obtiene la temperatura en grados Celsius dada la dirección única del sensor.

Devuelve la temperatura en un tipo de dato float.

```
Serial.print("Temperatura sensor congelador: ");
float temperaturaCongeladorCelsius = sensorDS18B20.getTempC(sensorCongelador);
Serial.print(temperaturaCongeladorCelsius);
Serial.println(" C");
```

```
float getTempF(const uint8_t*)
```

Obtiene la temperatura en grados Fahrenheit dada la dirección única del sensor. Devuelve la temperatura en un tipo de dato float.

```
Serial.print("Temperatura sensor congelador: ");
float temperaturaCongeladorFahrenheit =
sensorDS18B20.getTempC(sensorCongelador);
Serial.print(temperaturaCongeladorFahrenheit);
Serial.println(" F");
```

```
float getTempCByIndex(uint8_t)
```

Obtiene la temperatura en grados Celsius dado el índice del sensor dentro del bus 1-Wire. Devuelve la temperatura en un tipo de dato float.

```
Serial.print("Temperatura sensor 0: ");
float temperaturaCongeladorCelsius = sensorDS18B20.getTempCByIndex(0);
Serial.print(temperaturaCongeladorCelsius );
Serial.println(" C");
```

```
float getTempFByIndex(uint8_t)
```

Obtiene la temperatura en grados Fahrenheit dado el índice del sensor dentro del bus 1-Wire. Devuelve la temperatura en un tipo de dato float.

```
Serial.print("Temperatura sensor 0: ");
float temperaturaCongeladorFahrenheit = sensorDS18B20.getTempCByIndex(0);
Serial.print(temperaturaCongeladorFahrenheit);
Serial.println(" F");
```