

Mesurer la température avec une sonde DS18B20 (1-wire)

<https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html>

<http://www.framboise314.fr/mesure-de-temperature-1-wire-ds18b20-avec-le-raspberry-pi/>

<https://fr.wikipedia.org/wiki/HVAC>

Pour mesurer une température on peut utiliser différents types de capteur

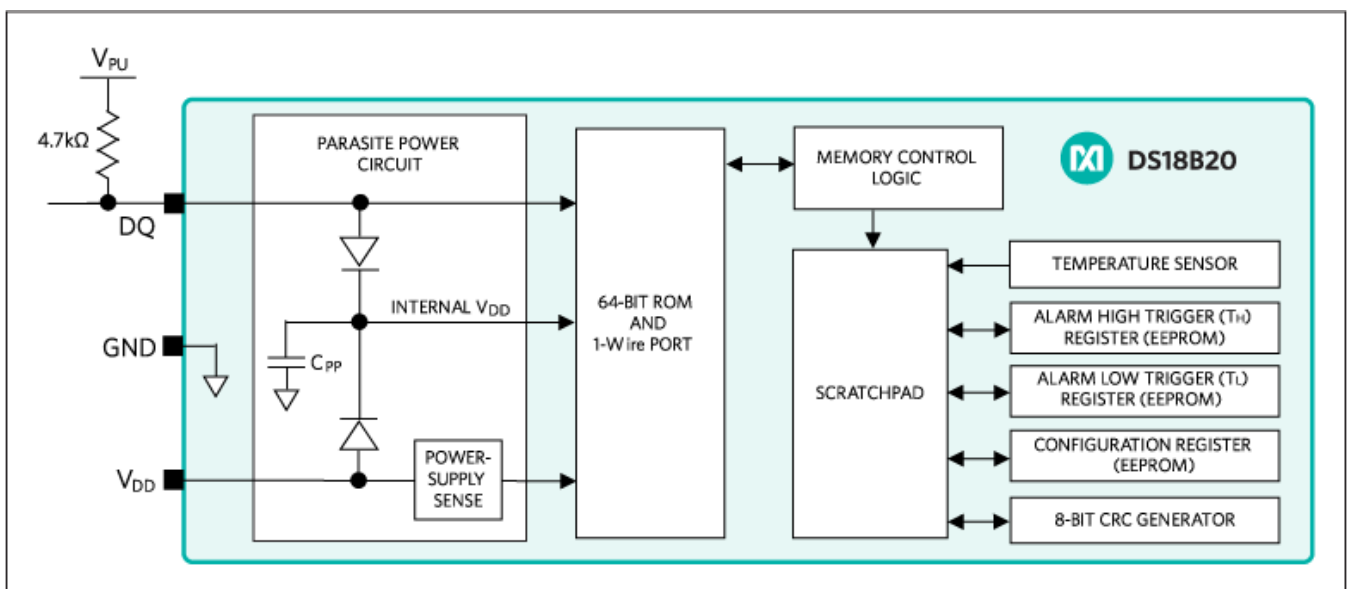
- un capteur analogique qui fournit une valeur (résistance/courant/tension proportionnelle à la température de la sonde)
- un capteur numérique qui fournit directement la mesure sous forme numérique.

Le capteur analogique n'est pas directement utilisable sur le Raspberry Pi, il délivre une tension/un courant analogique et nécessite la mise en œuvre d'une carte de mesure avec convertisseur analogique => digital (ADC ou CAN).

Parmi les capteurs numériques, une famille est particulièrement adaptée au Raspberry Pi, c'est la série 1-wire, dont les capteurs utilisant un seul fil pour transmettre l'information (nous verrons qu'il en faut... trois tout de même). Le Raspberry Pi (en fait Linux) embarque nativement le pilote capable de gérer cette famille de composants.

Comment ça marche ?

DS18B20



Le thermomètre numérique DS18B20 fournit des mesures de température en degrés Celsius avec une valeur codée de 9 à 12 bits. Sa gamme de mesure s'étend de -55°C à $+125^{\circ}\text{C}$ avec une précision de $\pm 0,5^{\circ}\text{C}$.

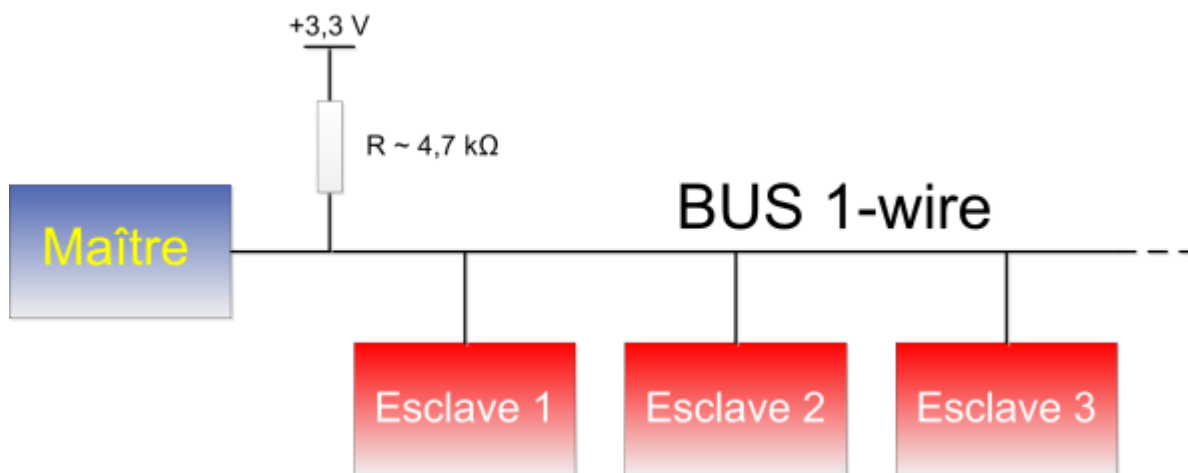
Il dispose d'une fonction d'alarme avec des points de déclenchement haut et bas non volatiles et programmables par l'utilisateur.

Chaque DS18B20 a un numéro de série unique sur 64 bits, ce qui permet à plusieurs DS18B20 de fonctionner sur le même bus 1-Wire. Il est ainsi simple d'utiliser un Raspberry Pi pour contrôler de nombreux DS18B20 répartis sur une grande surface. Parmi les applications pouvant bénéficier de cette fonctionnalité, on trouve

- les commandes environnementales HVCA (chauffage – ventilation – climatisation)
- les systèmes de surveillance de la température à l'intérieur des bâtiments
- la surveillance d'équipement ou de machines
- les systèmes de surveillance et de contrôle de processus industriel.

Le bus 1-wire

Le bus 1-wire est basé sur une architecture maître-esclave. Le maître est l'équipement qui contrôle le bus, interroge les périphériques, ou leur envoie des ordres.



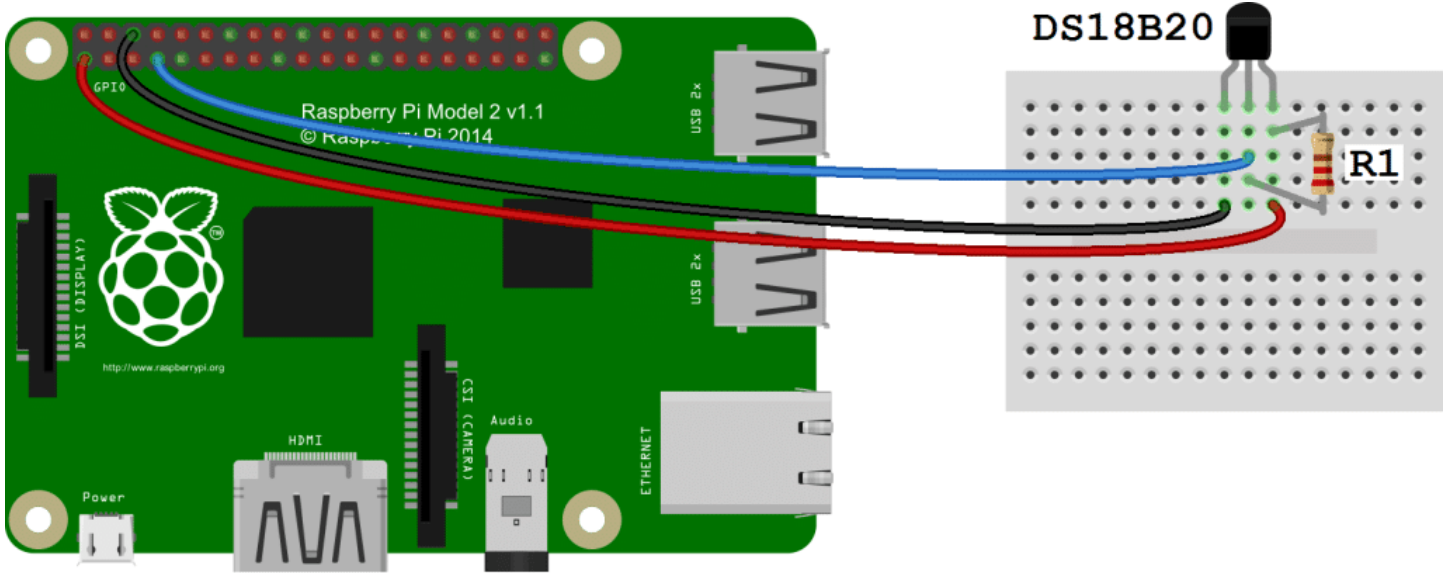
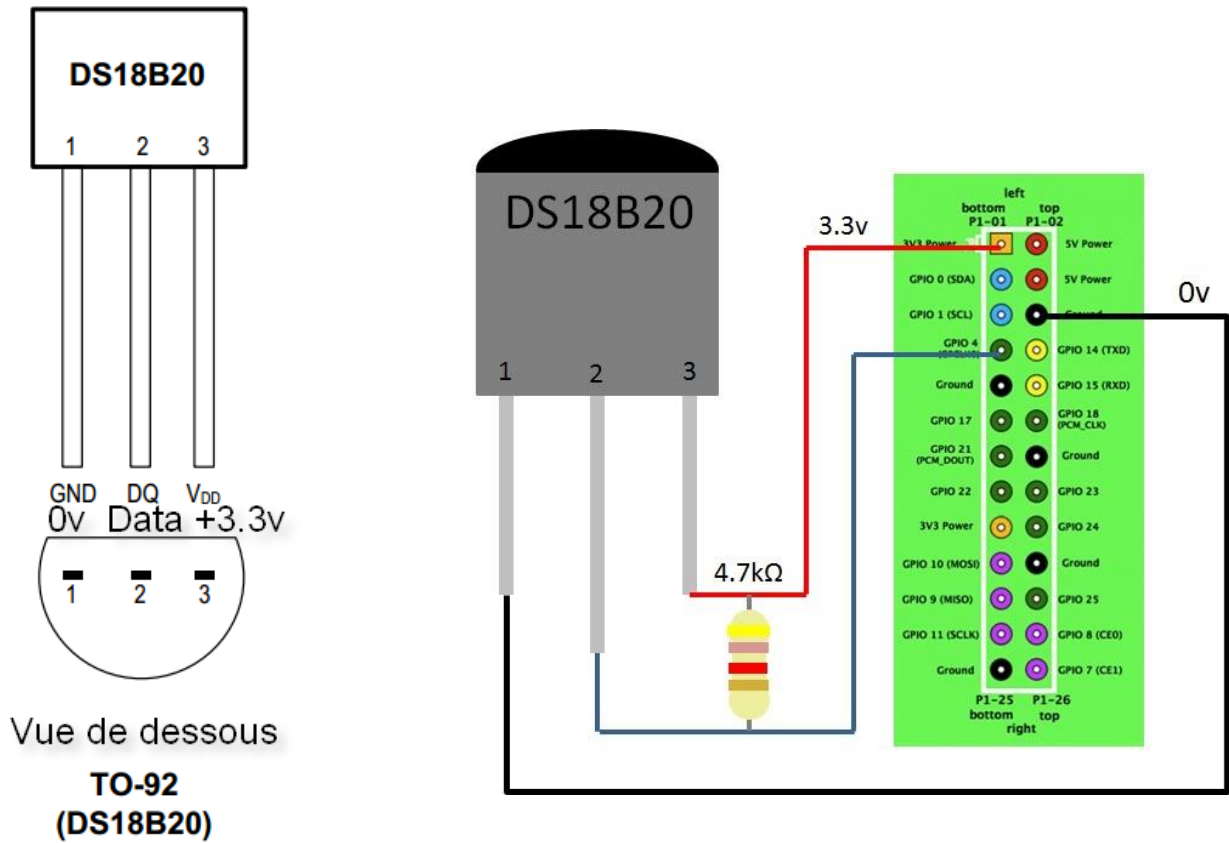
Dans notre cas, le maître sera le Raspberry Pi et il n'y aura qu'un esclave, notre capteur de température DS18B20.

Sur le schéma ci-dessus, on voit que le bus 1-wire est relié au +3,3 V par une résistance de tirage de 4,7 kΩ. En fonction de la longueur des fils de liaison, il sera parfois nécessaire d'ajuster sa valeur. Tous les appareils sont branchés en parallèle sur le bus et c'est en mettant (ou pas) leur sortie à la masse qu'ils envoient les données au maître.

Le maître initialise le bus en forçant un zéro pendant plus de 480 μs. Les esclaves lui répondent en mettant eux aussi leur sortie à zéro pendant un certain temps, indiquant ainsi leur présence.

Le maître déclenche une lecture dite ROM et l'esclave renvoie son identifiant unique, gravé lors de la fabrication. Chaque identifiant a une longueur de 48 bits (8 octets) encadrés par un octet indiquant le type de matériel (ici le type est : sonde de température = 28h) et un octet de [CRC](#) qui permet au maître de vérifier qu'il a correctement reçu les informations d'identification du composant. Les 48 bits permettent d'individualiser à peu près 280 000 milliards de composants...

Branchement du DS18B20 sur le GPIO



fritzing

Le branchement se fait sur l'alimentation +3,3v et Masse. Une résistance de tirage (4,7K) est connectée entre +3,3v et broche de sortie des data (pull-up).

Utilisation de base du DS18B20

Vérification manuelle du bon fonctionnement

Avant de commencer les tests, il faut activer le bus 1-wire dans la fenêtre de configuration du Raspberry Pi

Lorsque le bus 1-wire est activé on peut vérifier dans le dossier `/sys/bus/w1/devices` que le système a bien reçu un identifiant unique envoyé par notre DS18B20

```
pi@raspberrypi:~ $ cd /sys/bus/w1/devices
pi@raspberrypi:/sys/bus/w1/devices $ ls -al
total 0
drwxr-xr-x 2 root root 0 oct. 12 14:11 .
drwxr-xr-x 4 root root 0 oct. 12 14:10 ..
lrwxrwxrwx 1 root root 0 oct. 12 14:11 28-000006dfa7e5 lrwxrwxrwx 1 root
root 0 oct. 12 14:10 w1_bus_
```

Ici le DS18B20 a pour numéro de série **28-000006dfa7e5**

Le premier octet **28** est le code famille du DS18B20

Les 6 octets suivants **000006dfa7e5** donnent le numéro de série du composant

Un octet non affiché contient un code de vérification de type CRC qui permet de s'assurer que le N° de série a bien été reçu.

Lecture de la température en ligne de commande

Ceci permet de confirmer le bon fonctionnement de l'ensemble capteur/système. S'il est impossible de lire la température en ligne de commande, il est inutile de continuer. Il faut trouver pourquoi ça ne fonctionne pas !

```
pi@raspberrypi:/sys/bus/w1/devices $ cd 28-000006dfa7e5
pi@raspberrypi:/sys/bus/w1/devices/28-000006dfa7e5 $ ls
driver id name power subsystem uevent w1_slave
```

La température mesurée se trouve dans `w1_slave` :

```
pi@raspberrypi:/sys/bus/w1/devices/28-000006dfa7e5 $ cat w1_slave
c5 01 4b 46 7f ff 0b 10 16 : crc=16 YES
c5 01 4b 46 7f ff 0b 10 16 t=28312
```

Il faut diviser la valeur par 1000 pour connaître la température relevée : **28,312 °C**.

Vu la précision de $\pm 0,5^\circ\text{C}$ on pourra arrondir la valeur.

Autre mesure :

```
pi@raspberrypi:/sys/bus/w1/devices/28-000006dfa7e5 $ cat w1_slave
42 01 4b 46 7f ff 0e 10 ab : crc=ab YES
42 01 4b 46 7f ff 0e 10 ab t=20125
```

Cette fois la température mesurée est de **20 °C**

On peut bien entendu aller plus loin en automatisant l'extraction de la valeur contenue dans le fichier :

```
pi@raspberrypi:~ $ find /sys/bus/w1/devices/ -name "28-*" -exec cat {}/w1_slave \; | grep "t=" | awk -F "t=" '{print $2/1000}'  
21.625
```

Je vous laisse décortiquer cette commande ☺

Mise en pratique

Dans chacun des programmes il faudra indiquer le numéro unique de votre capteur de température.

Lecture de la température

Ecrire un script *lit_temp_GEII_2018.sh* qui lit la température une fois par seconde et l'affiche sur l'écran.

Détection de seuil

Ce script *lit_temp_seuil_GEII_2018.sh* simule la détection d'une surchauffe, et peut servir à mettre en route un système de refroidissement.

Ecrire un script qui lit la température une fois par seconde et l'affiche sur l'écran. Lorsque la température dépasse une valeur déterminée (à régler en fonction de la température du jour), allumer une LED. Eteindre la LED lorsque la température est redescendue d'au moins un degré (hystérésis).

Ouverture de serre

Ce script simule l'ouverture d'un panneau d'aération de serre en fonction de la température

Ecrire un script qui lit la température une fois par seconde et l'affiche sur l'écran. Lorsque la température dépasse une première valeur déterminée (à régler en fonction de la température du jour), un servomoteur positionné au départ en butée inférieure, passe en position médiane. Lorsque la température dépasse une deuxième valeur déterminée (à régler en fonction de la température du jour), le servomoteur passe en butée supérieure. Le servomoteur revient en arrière lorsque la température est redescendue d'au moins un degré dans les 2 cas (hystérésis).

Par exemple :

#1 < à 23°C

#2 > à 23°C < 25°C

#3 > à 25°C

Pour #1 lorsque la température dépasse 23°C on ouvre en position #2 et la température de fermeture (retour à #1) passe à 22°C

Pour #2 lorsque la température dépasse 25°C on ouvre en position #3 et la température de fermeture (retour à #2) passe à 24°C

Nota : création de nom de variable par concaténation `$test = ${'test'}.${toto}`

