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

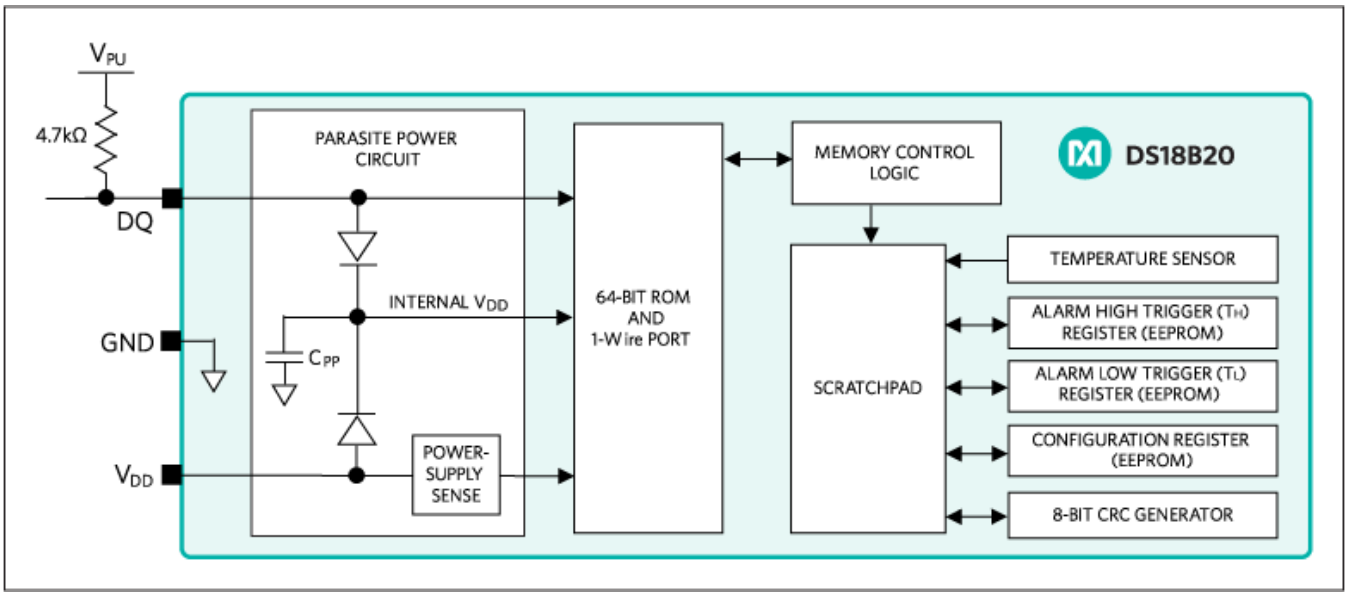
- un capteur analogique qui fournit une valeur (résistance/courant/tension proportionnelle à la température de la sonde)
- un capteur numérique qui fournit directement la mesure sous forme numérique.

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°C avec une précision de ±0,5°C.

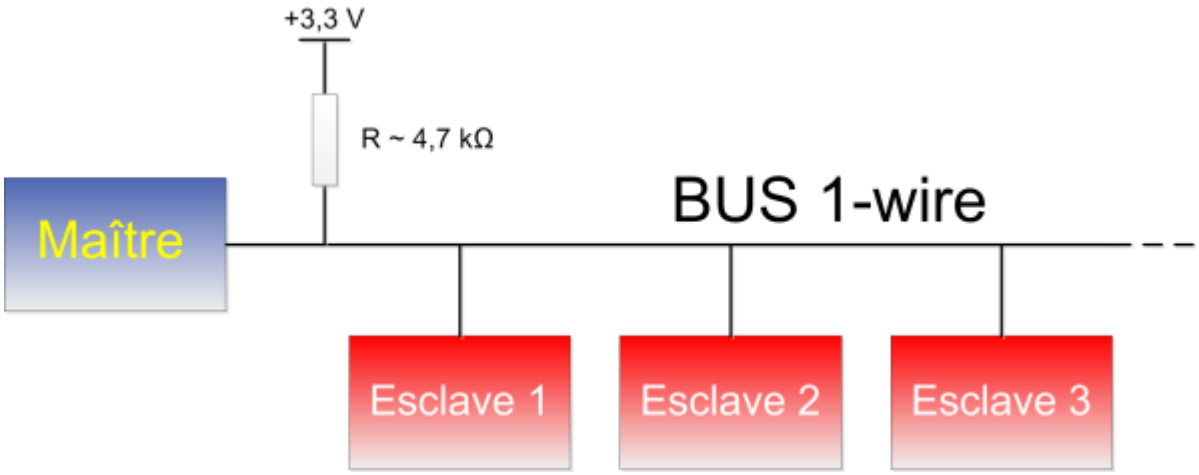
Il dispose d'une fonction d'alarme avec des points de déclenchement haut et bas non volatiles et programmables par l'utilisateur.

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.



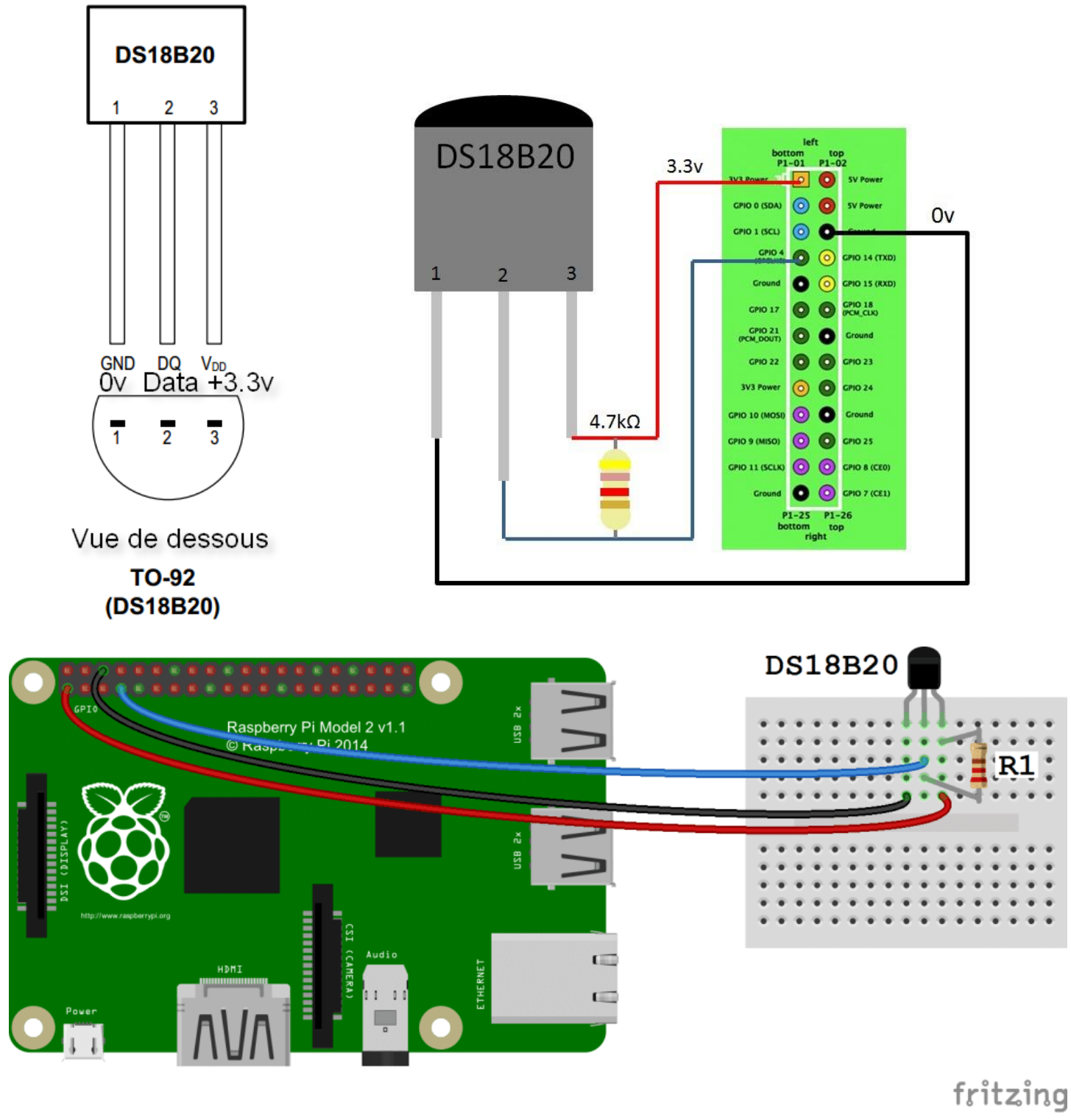
Dans notre cas, le maître sera le Raspberry Pi et il n'y aura qu'un esclave, notre capteur de température DS18B20.

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 mettent 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



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).

Vérification 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 connaitre la température relevée : **28,312 °C**.
Vu la précision de ±0,5°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 ☺

Utilisation de wiringPi avec Qt Creator

Objectif : Ecrire un programme avec Qt pour lire la température en mode console, l'afficher puis fermer le programme

Créer un nouveau projet en mode console : ***temperature***
Ajouter la bibliothèque wiringPi à ***temperature.pro***
Saisir le programme ***main.cpp***

```
#include <QCoreApplication>
#include <iostream>
#include <wiringPi.h>
#include <ds18b20.h>
#include <qdebug.h>

int main(int argc, char *argv[])
{
int temp;

    QCoreApplication a(argc, argv);

    wiringPiSetupSys();
    ds18b20Setup (100, "000006dfa7e5") ;
    qDebug("Mesure");
    temp = analogRead (100) ;
    qDebug("Mesure finie");
    printf("Temperature : %d \n", temp);
    std::cout << "Temperature : " << temp << "\n";
    qDebug("Affichage temp fini");

    return 0;
}
```

Quelques explications :

```
#include <ds18b20.h>
#include <qdebug.h>
```

Les lignes permettent de gérer les capteurs de température et d'envoyer des messages de debug qu'on pourra supprimer quand le programme sera au point.

ds18b20Setup(100, "000006dfa7e5")
100 est un numéro d'ID que vous donnez au capteur. Il est quelconque mais doit être plus grand que 64 et être utilisé une seule fois dans le programme.
000006dfa7e5 est le numéro de série du capteur que vous avez connecté et qui apparait dans `/sys/bus/w1/devices`

temp = analogRead (100)

Cette fonction lit le capteur dont l'ID est mentionné et retourne ***la température multipliée par 10 !***

```
   printf("Temperature : %d \n", temp);
   std::cout << "Temperature : " << temp << "\n";
```

Montre les deux méthodes pour sortir une valeur sur la console : le printf() du C et le std ::cout de C++

return 0;
Permet de sortir du programme sans devoir utiliser CTRL C
Remplace ***a.exec()*** qui est une boucle d'***event*** qui écoute les évènements de l'application (clic souris, appuis sur des touches, déplacement de la souris, réception réseau ...) et ne rend la main que lorsqu'elle reçoit un évènement de fermeture comme CRTL C par exemple.

Fonctionnement du programme sur la console :

```
pi@raspberrypi:~/Projets/build-temperature-Desktop-Debug $ ./temperature
Mesure
Mesure finie
Temperature : 220
Temperature : 220
Affichage temp fini
```

Programme en C

On peut également écrire directement le programme ***temp.c*** en langage C, sans passer par Qt

```
#include <stdio.h>
#include <wiringPi.h>
#include <ds18b20.h>
int main (void)
{
int temp;
   wiringPiSetupSys();
   ds18b20Setup (100, "000006dfa7e5") ;

   temp = analogRead (100) ;
   printf("Temperature: %d \n", temp);

 return 0 ;
}
```

Puis le compiler :
```
pi@raspberrypi:~/Projets/C $ gcc -Wall -o temp temp.c -lwiringPi
```

Et enfin exécuter le fichier ***temp*** :
```
pi@raspberrypi:~/Projets/C $ ./temp
Temperature: 221
```

Comparatif de la taille des fichiers :

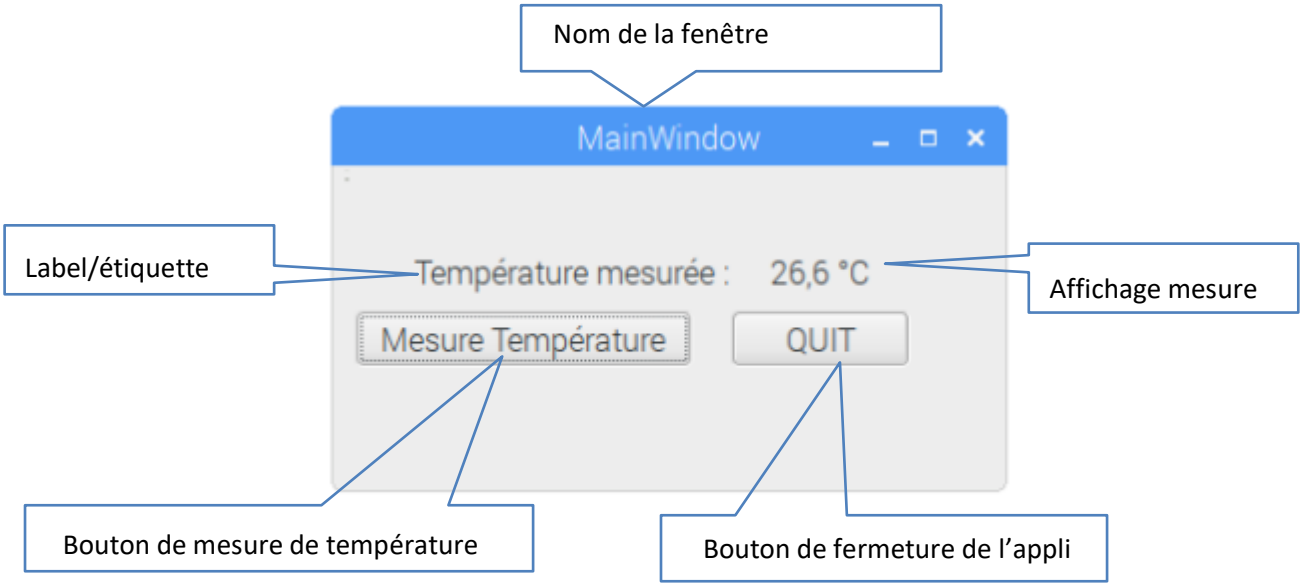
Le programme ***temp*** compilé en C a une taille de 3Ko
```
-rwxr-xr-x  1 pi pi 8364 oct.  13 10:50 temp
```

Le programme ***temperature*** créé par Qt a une taille de 30Ko
```
-rwxr-xr-x  1 pi pi 30428 oct.  13 11:06 temperature
```

Ces tailles sont dans un rapport de 10. Même si les capacités mémoire des machines actuelles dépassent largement ces valeurs, il faut considérer que dans certaines configurations de matériel embarqué il est important de réduire la taille des exécutables.

Mise en pratique

1- Créer un programme graphique mesurant la température quand on clique sur un bouton. La température s'affiche dans la fenêtre. Un bouton « QUIT » permet de fermer « proprement » l'application.



Pour information le programme en mode graphique occupe plus de 600 Ko

```
-rwxr-xr-x  1 pi pi 654872 oct.  13 15:22 temp_UI
```

2- Créer un programme graphique comme ci-dessus. Le bouton Mesure Température va cette fois démarrer la lecture et l'affichage de la température. L'opération se répétera ensuite automatiquement toutes les 5 secondes. Un bouton « QUIT » permet de fermer « proprement » l'application.