RASPBERRY PI

Un PC pédagogique pour (deuxième partie)

BRUNO GUILBERT [1]

Dans la première partie de l'article (voir* Technologie *n° 189, janvier-février 2014), nous présentions la carte Raspberry Pi et l'installation de son système d'exploitation Raspbian. Dans cette deuxième partie, nous vous proposons d'utiliser notre carte RPi dans une application concrète.

mots-clés
hardware, projet, système d'exploitation programmation

L'application relative à la régulation de température permet de poser les bases d'une exploitation pédagogique de la carte à un niveau BTS. La régulation de la température d'un appareil de chauffage électrique peut se faire de plusieurs façons : soit par thermostat, soit par régulateur PID (proportionnel intégral dérivé), ou encore par procédé thermocyclique. Nous avons choisi la plus simple : la régulation en TOR (tout ou rien) par thermostat (un prochain article abordera un autre type de régulation qui permet d'obtenir plus de précision entre la consigne et la température réelle).

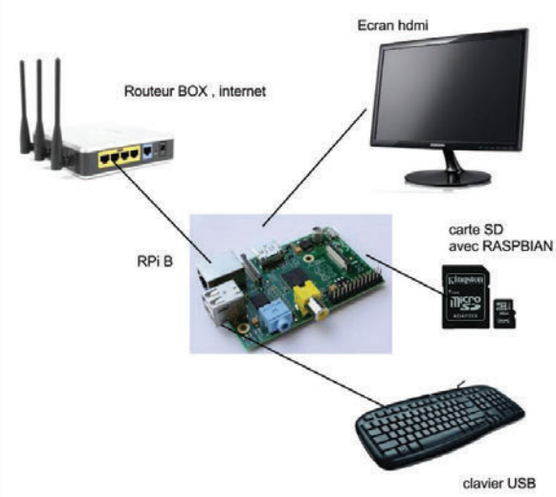
Le choix de cette application pédagogique concorde avec le référentiel du BTS SN dans le cadre d'interventions que le futur technicien pourra effectuer soit dans des phases de test, soit dans des phases de mise au point de systèmes numériques.

Mise en place du RPi sur le réseau Ethernet

Une fois notre problème posé, il nous faut la carte et la version Debian pour Raspberry Pi, nommée Raspbian. Ce système d'exploitation a pu être mis en place à l'aide de l'outil Noobs, un assistant qui permet d'installer un ou plusieurs systèmes d'exploitation sur une carte mémoire de type SD qui se logera dans la RPi.

L'accès au RPi se fait par son interface Ethernet après y avoir paramétré une IP fixe. Pour réaliser cette opération, il faut un écran et un clavier directement connectés au RPi 1. Pour des raisons d'ergonomie, nous le placerons dans un boîtier de protection. Il suffira ensuite de l'alimenter et de le brancher au réseau Ethernet.

La configuration peut se faire aussi bien à domicile que dans une salle de cours. Nous partons du principe



1 La configuration de départ

que le réseau local est de type Ethernet, constitué d'un routeur ou bien d'un modem/routeur tel qu'une box. Il faut connecter le RPi au réseau local en RJ45. Depuis l'écran connecté sur le RPi, nous nous connectons de la manière suivante :

```
login pi
passwd raspberry
```

Pour éditer le fichier de configuration du réseau, il suffit d'entrer la commande :

```
nano /etc/network/interfaces
```

Le fichier ainsi édité est le suivant :

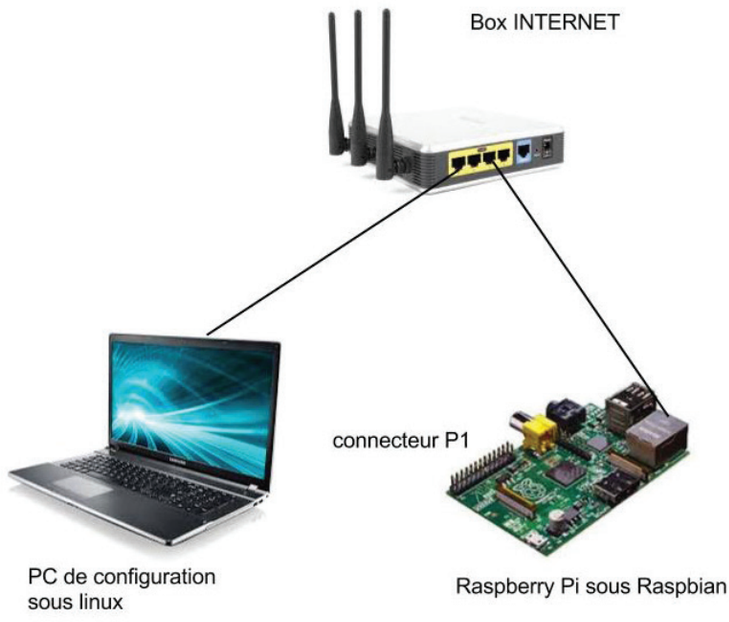
```
auto lo
iface lo inet loopback
iface eth0 inet static

address 192.168.0.60
netmask 255.255.255.0
gateway 192.168.0.254

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

[1] Professeur en STS IRIS au lycée Jacquard de Paris (75019).

moins de 100 euros



2 La configuration de travail

La ligne, notée « address », contient l'adresse IP fixe que nous avons choisie pour le RPi, dans notre cas il s'agit de l'adresse 192.168.0.60. Attention, selon l'opérateur, le type d'adresse ne sera pas le même (par exemple : 192.168.0.60 pour un abonné Free et 192.168.1.60 pour un abonné Orange). Après un redémarrage, l'adresse du RPi dans le réseau Ethernet sera donc 192.168.0.60, le masque de sous-réseau sera 255.255.255.0, et la passerelle 192.168.0.254. Selon les box, les passerelles utilisées peuvent être 192.168.1.1 ou 192.168.0.254.

Après le redémarrage via la commande sudo reboot, le RPi sera accessible depuis le réseau sur n'importe quel ordinateur muni d'un système d'exploitation de type Linux permettant de saisir la commande SSH. À ce stade, l'écran et le clavier dédié initialement connectés ne sont plus nécessaires **2**.

Une fois la carte reliée au réseau local, il faut la mettre sous tension, attendre un peu, puis allumer l'ordinateur. Enfin, il faut la commande suivante dans un terminal (voir *Technologie* n° 189, janvier-février 2014) :

```
ssh pi@192.168.0.60 ,
```

Le résultat obtenu est le suivant :

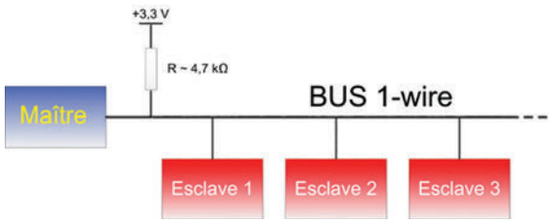
```
bruno@bruno-HP-ProDesk-600-G1-TWR:~$
ssh pi@192.168.0.60
The authenticity of host '192.168.0.60
(192.168.0.60)' can't be established.
ECDSA key fingerprint is
13:0b:b4:f1:9e:5e:3b:16:5a:85:d0:4e:7e:15:66:8b.
Are you sure you want to continue connecting
(yes/no)? yes
Warning: Permanently added '192.168.0.60'
(ECDSA) to the list of known hosts.
pi@192.168.0.60's password:
Linux raspberrypi 3.12.28+ #709 PREEMPT
Mon Sep 8 15:28:00 BST 2014 armv6l
The programs included with the Debian GNU/Linux
system are free software;
the exact distribution terms for each program
are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with
ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Sep 9 08:10:46 2014
pi@raspberrypi ~ $
```

Dorénavant, nous disposons d'un RPi accessible à distance depuis un PC connecté sur le même réseau local, condition nécessaire pour recevoir un capteur de température.

Mise en place du capteur de température

Nous avons choisi un capteur de température standard, le plus couramment utilisé, à savoir le DS18B20 **3**. Cette sonde de température offre une précision de +/-0,5 °C sur la plage de -10 °C à +85 °C, précision suffisante pour une application de chauffage domestique comme dans notre étude.

3 La sonde DS18B20 sous ses deux formes



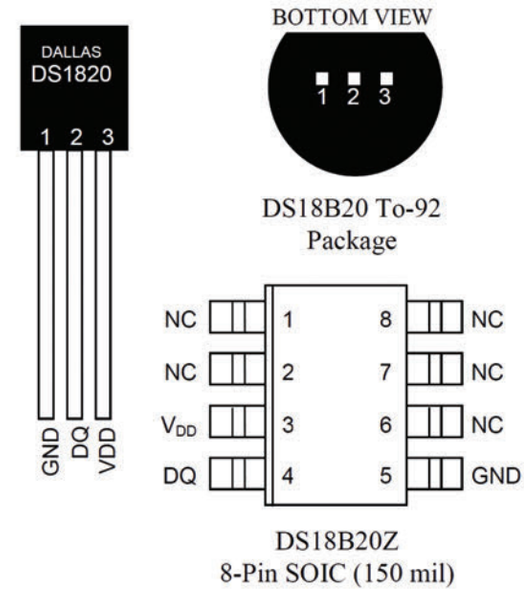
4 Le Bus 1 Wire (les alimentions ne sont pas représentées)

La sensation de froid ou de chaud apparaît à condition que les variations de température soient au moins de 6 °C par minute. Si la variation de température est plus lente, l'écart thermique peut devenir très important avant que nous ne ressentions un changement de température. Notre régulateur ne devra pas provoquer ce genre de variation.

Pour exploiter cette sonde, nous la placerons sur un bus 1-Wire 4 qui servira uniquement de support au transport des données. Il est câblé sur la ligne GPIO 4. Deux fils supplémentaires sont nécessaires pour alimenter le capteur, ce qui implique l'usage de trois fils en tout. Préférer la version sonde plutôt que la version composant électronique, mais, pour ceux qui le souhaitent, le brochage des composants électroniques est présenté en 5.

Pour un câblage en version sonde, le fil rouge est le VDD (+3.3V), le noir GND et le blanc DQ In/Out.

Notre RPi pilote la sonde ; elle prendra ici le rôle de maître. Le capteur de température, quant à lui, jouera le rôle d'esclave. Ce dernier est prévu pour fonctionner de 3.0 à 5.0 volt le RPi fonctionnant en 3.3 volt, son alimentation convient donc pour le capteur. La résistance de 4.7 kΩ préconisée par le Dallas (le constructeur du DS18B20) est une « résistance de niveau » ou un « rappel au niveau haut » (en langage d'électronicien : une « résistance de tirage au niveau haut » ou « de Pull-UP ») 4.



PIN DESCRIPTION

GND	- Ground
DQ	- Data In/Out
V_{DD}	- Power Supply Voltage
NC	- No Connect

5 Le brochage des boîtiers To 92 et SOIC

Mise en place du « module » logiciel pour gérer la sonde

Un module est un morceau de programme qui s'insère au noyau pour permettre la gestion d'un composant, en l'occurrence la sonde DS18B20. Ce module peut aussi se nommer pilote, ce qui, par analogie, s'apparente à un driver d'imprimante.

Pour que le noyau Linux sache gérer le composant 1-wire DS18B20, il faut lui adjoindre les deux modules : w1-gpio pour la gestion du bus 1-wire, et w1-therm pour la gestion des composants de la famille 28h (famille des capteurs de température). Ces modules existent nativement dans le système de fichier du RPi, mais ils ne sont pas chargés.

En figure a la procédure pour charger un module avec la fonction *modprobe*.

Le module w1_gpio a bien été ajouté. Pour ajouter le deuxième module :

```
pi@raspberrypi ~ $ sudo modprobe -a w1_therm
```

Dans le cas où un module doit être déchargé, l'option -r peut être utilisée :

```
pi@raspberrypi ~ $ sudo modprobe -r w1_therm
```

```
pi@raspberrypi ~ $ sudo modprobe -a w1-gpio
pi@raspberrypi ~ $ lsmod
```

```
Module                  Size Used by
w1_gpio                 2751 0
wire                    25349 1 w1_gpio
cn                      4780 1 wire
snd_bcm2835             19496 0
snd_soc_bcm2708_i2s     6210 0
regmap_mmio             2818 1 snd_soc_bcm2708_i2s
8192cu                  8192cu  550797 0
snd_soc_core            127849 1 snd_soc_bcm2708_i2s
snd_compress            8259 1 snd_soc_core
regmap_i2c              1661 1 snd_soc_core
snd_pcm_dmaengine       5505 1 snd_soc_core
regmap_spi              1913 1 snd_soc_core
snd_pcm                 83845 3 snd_bcm2835,snd_soc_core,snd_pcm_dmaengine
snd_page_alloc          5132 1 snd_pcm
snd_seq                 55484 0
snd_seq_device          6469 1 snd_seq
snd_timer               20998 2 snd_pcm,snd_seq
leds_gpio               2079 0
led_class               4118 1 leds_gpio
snd                     62252 7 snd_bcm2835,snd_soc_core,snd_timer,snd_
.snd_compress           pcm,snd_seq,snd_seq_device
pi@raspberrypi ~ $
```

a

Notre application devant être autonome, il faut automatiser le chargement des deux modules au démarrage du RPi. Pour cela, il suffit d'ajouter dans le dossier « /etc/modules » les modules w1-gpio et w1-therm, ce qui se traduit par l'exécution des commandes suivantes b :

```
sudo nano modules
```

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
w1-gpio pullup=1
w1_therm
```

b

Les modifications seront effectives au redémarrage de la machine. Après un redémarrage c :

```
pi@raspberrypi /sys/bus/w1/devices/
28-000004efd556 $ ls
```

```
driver id name power subsystem uevent w1_slave
pi@raspberrypi /sys/bus/w1/devices/28-000004efd556 $ more w1_slave
8f 01 4b 46 7f ff 01 10 14 : crc=14 YES
 8f 01 4b 46 7f ff 01 10 14t=24937
pi@raspberrypi /sys/bus/w1/devices/28-000004efd556 $
```

c

Le *device* interrogé renvoie la température sur la variable « t ». Dans l'exemple c : t = 24937, ce qui correspond à 24,937 °C.

Pour extraire cette température de toutes les données relatives au capteur de température, on utilise le petit script shell suivant :

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

Pour éviter de retaper la ligne de commande et pour l'exécuter à loisir, il suffit de la placer dans un fichier script nommé « lecture_temp.sh » :

```
nano lecture_temp.sh
```

Pour quitter l'éditeur, taper sur la touche Ctrl-X, en ayant sauvegardé au préalable.

Si nous plaçons la sonde proche d'une source de chaleur en répétant notre script à intervalle régulier, nous recueillerons les données qui indiqueront une augmentation de la température :

```
pi@raspberrypi ~ $ ./lecture_temp.sh
24.937
pi@raspberrypi ~ $ ./lecture_temp.sh
25.062
pi@raspberrypi ~ $ ./lecture_temp.sh
28.437
```

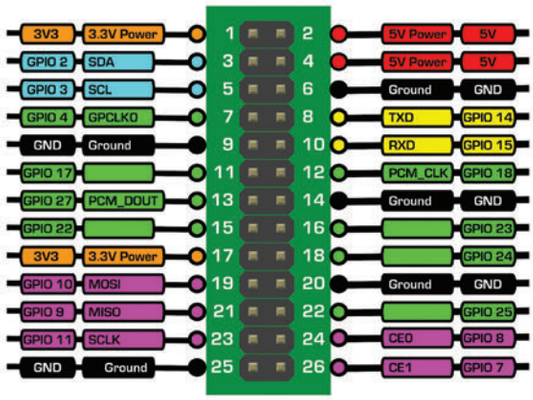
Mise en place de la partie puissance pour le radiateur

Pour piloter le radiateur, il est nécessaire d'ajouter un relais électromagnétique 6 qui permet d'obtenir une isolation galvanique entre la partie commande (signaux) de la partie radiateur (forte puissance). Sur la partie commande de ce relais, il suffit d'apporter une alimentation 5 volt et le signal de commande, tous deux issus de la carte RPi. Sur la partie puissance, on apporte le 230 volt sur le commun, et on peut choisir le type de logique de commande positive ou négative : NC Normalement Connecté, NO Normalement Ouvert. Nous choisissons le GPIO 0 pour la commande du relais. La broche notée IN du relais sera donc reliée à la broche BCM 17 / GPIO 0.

Mise en place logiciel

Pour piloter les GPIO (General Purpose Input Output) du RPi, la méthode la plus simple consiste à installer une surcouche logicielle appelée WiringPi, disponible gratuitement à l'adresse suivante : http://wiringpi.com/download-and-install/

6 Un relais de puissance pour RPi/Arduino



7 Le connecteur P1 de la RPi vu du dessus

Une fois le fichier téléchargé, il faut le décompresser dans l'espace de travail du RPi en suivant la procédure figurée en d (test inclus).

On peut y noter les états des broches au moment du test.

La fonction GPIO permet d'écrire, de lire et de décider de la configuration en entrée ou en sortie de chaque broche 7 et d. Il est possible de tester le relais en ligne de commande (shell).

Pour obtenir une aide sur l'utilisation de la commande GPIO, saisir :

```
man gpio
```

Pour mettre la ligne en sortie afin de commander le relais, saisir :

```
gpio mode 0 output
```

Pour activer le relais (remarque : le 0 correspond au GPIO 17 7) :

```
gpio write 0 1
```

Pour désactiver le relais :

```
gpio write 0 0
```

Le câblage de notre prototype 8 est tel que :

– GPIO 4 : est réservé pour le bus 1-Wire pour la sonde DS18B20 ;

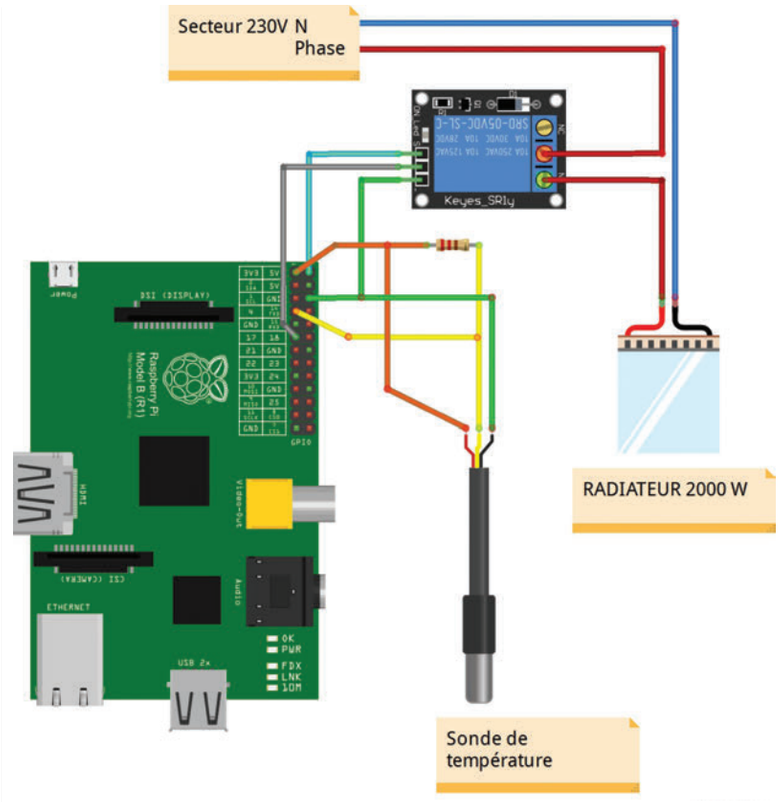
– GPIO 17 : commande le relais, qui pilote le radiateur.

Attention : le câblage de la partie puissance doit respecter les prescriptions de la NF C 15-100. La puissance du radiateur ne devra pas dépasser les caractéristiques du relais, 2 500 watt dans notre cas. On peut constater le câblage terminé 9.

Réalisation d'une régulation de température avec une consigne fixe

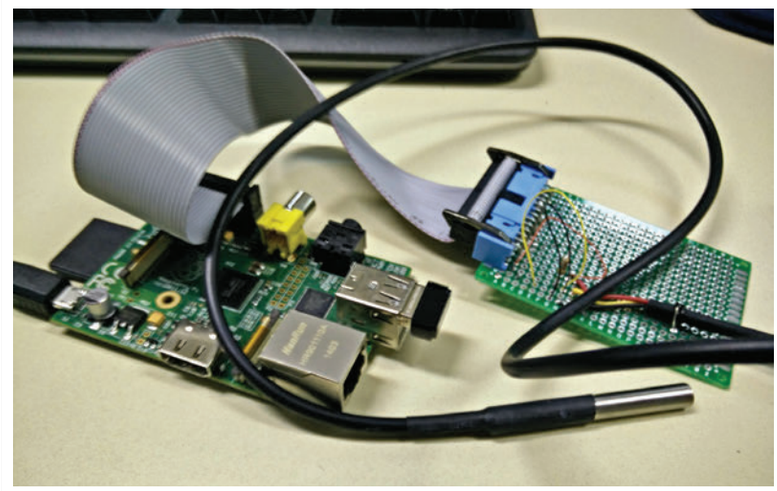
Maintenant que notre carte est équipée de la sonde et qu'elle peut recevoir des données de température, nous allons programmer un système simple de régulation de la température en TOR avec une hystérésis 10, ce qui permettra d'éviter une trop forte sollicitation de notre relais de puissance au voisinage de la consigne. L'hystérésis choisie est de un demi-degré.

En encadré (p. 46), le codage du programme principal en C écrit dans le fichier nommé « regul.c », les commentaires du programme sont en bleu.



8 La proposition de câblage (réalisé avec le logiciel gratuit Fritzing)

En encadré (p. 47), le fichier *makefile* pour fabriquer notre projet. Nous utilisons l'outil *make* pour générer notre fichier exécutable nommé *regul*. Attention le caractère « → » dans le texte de *makefile* représente une tabulation qu'il convient de respecter. Il faut placer dans le répertoire « /home/pi/regul » les deux fichiers : *regul.c* et *makefile*, puis, pour compiler le tout, taper la commande : *make*. Cette commande dans le répertoire générera, comme précisé précédemment, notre fichier exécutable : *regul*.



9 Le câblage du prototype

```
./build
  cd wiringPi
  make static
  sudo make install-static
pi@raspberrypi ~/wiringPi/wiringPi $ gpio readall
```

```
 +------+------+---------+------+---+-Model B2-+---+------+---------+-----+------+
 | BCM | wPi |  Name   | Mode | V |  Physical  | V | Mode |  Name   | wPi | BCM |
 +------+------+---------+------+---+-----++-----+---+------+---------+-----+------+
 |      |     |    3.3v |      |   |   1 ||   2 |   |      | 5v      |     |      |
 |    2 |   8 |   SDA.1 |   IN | 1 |   3 ||   4 |   |      | 5V      |     |      |
 |    3 |   9 |   SCL.1 |   IN | 1 |   5 ||   6 |   |      | 0v      |     |      |
 |    4 |   7 | GPIO. 7 |   IN | 1 |   7 ||   8 | 1 | ALT0 | TxD     | 15  | 14   |
 |      |     |      0v |      |   |   9 ||  10 | 1 | ALT0 | RxD     | 16  | 15   |
 |   17 |   0 | GPIO. 0 |   IN | 0 |  11 ||  12 | 0 |   IN | GPIO. 1 | 1   | 18   |
 |   27 |   2 | GPIO. 2 |   IN | 0 |  13 ||  14 |   |      | 0v      |     |      |
 |   22 |   3 | GPIO. 3 |   IN | 0 |  15 ||  16 | 0 |   IN | GPIO. 4 | 4   | 23   |
 |      |     |    3.3v |      |   |  17 ||  18 | 0 |   IN | GPIO. 5 | 5   | 24   |
 |   10 |  12 |    MOSI |   IN | 0 |  19 ||  20 |   |      | 0v      |     |      |
 |    9 |  13 |    MISO |   IN | 0 |  21 ||  22 | 0 |   IN | GPIO. 6 | 6   | 25   |
 |   11 |  14 |    SCLK |   IN | 0 |  23 ||  24 | 1 |   IN | CE0     | 10  | 8    |
 |      |     |      0v |      |   |  25 ||  26 | 1 |   IN | CE1     | 11  | 7    |
 +------+------+---------+------+---+-----++-----+---+------+---------+-----+------+
 |   28 |  17 | GPIO.17 |   IN | 0 |  51 ||  52 | 0 |   IN | GPIO.18 | 18  | 29   |
 |   30 |  19 | GPIO.19 |   IN | 0 |  53 ||  54 | 0 |   IN | GPIO.20 | 20  | 31   |
 +------+------+---------+------+---+-----++-----+---+------+---------+-----+------+
 | BCM | wPi |   Name  | Mode | V |  Physical  | V | Mode | Name    | wPi | BCM |
 +------+------+---------+------+---+-----++-----+---+------+---------+-----+------+
pi@raspberrypi ~/wiringPi/wiringPi $
```

d

Mise en place automatique du *daemon regul*

La mise en place de la régulation de la température se fera par un « service » (traduction du terme *daemon*) qui sera exécuté en permanence dans le RPi.

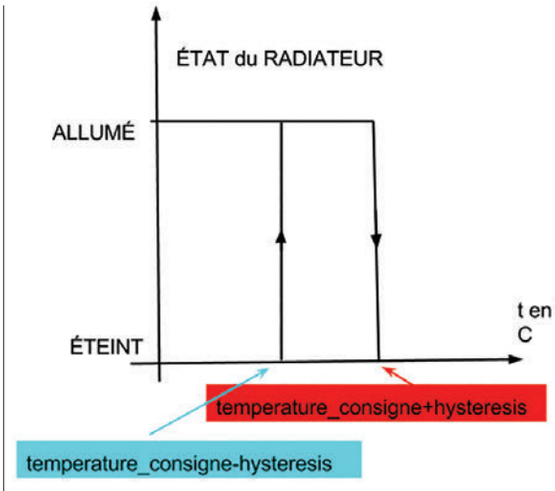
Si on lance manuellement notre programme, on peut vérifier qu'il fonctionne bien. Cependant, si on redémarre le RPi, le programme ne se lancera pas seul. Le *daemon* sert justement à lancer notre programme automatiquement au démarrage de notre RPi.

Le répertoire « /etc/init.d » e contient les scripts qui doivent être démarrés automatiquement au démarrage de notre RPi.

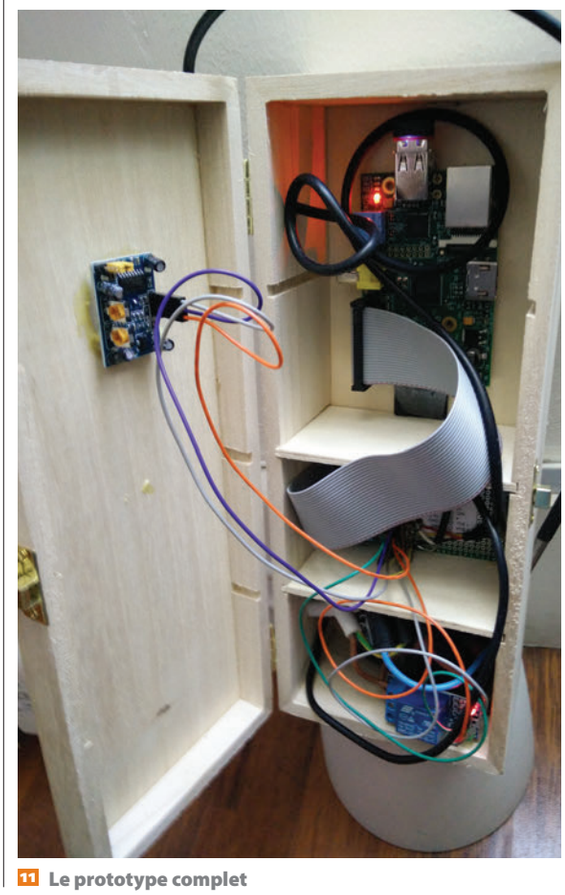
On placera donc dans ce répertoire un fichier script shell qui permettra d'exécuter notre application de régulation de température.

Rappel : il faut saisir la commande « nano/etc/init.d/regul.sh » en tant que *root* grâce à la commande sudo (voir *Technologie* n° 189, janvier-février 2014).

```
sudo nano /etc/init.d/regul.sh
```



10 Le cycle d'hystérésis



11 Le prototype complet

```
pi@raspberrypi /etc/init.d $ ls
alsa-utils              checkroot.sh    hostname.sh     lightdm              mountnfs.sh  plymouth-log  reboot     single       umountfs
apache2                 console-setup   hwclock.sh      motd                 mtab.sh      procps        regul.sh   skeleton     umountnfs.sh
bootlogs                cron            ifplugd         mountall-bootclean.sh mysql       raspi-config  rmnologin  ssh          umountroot
bootmisc.sh             dbus            kbd             mountall.sh          networking   rc            rpcbind    sudo         urandom
cgroup-bin              dphys-swapfile  keyboard-setup  mountdevsubfs.sh     nfs-common   rc.local      rsync      triggerhappy x11-common
checkfs.sh              fake-hwclock    killprocs       mountkernfs.sh       ntp          rcS           rsyslog    udev
checkroot-bootclean.sh  halt            kmod            mountnfs-bootclean.sh plymouth     README        sendsigs   udev-mtab
pi@raspberrypi /etc/init.d $
```

e

Le codage du programme principal en C

```
   /* Régulateur de température RPi V1.0 */
   /* (c) 2015 Bogt */
   /* inclusion des librairies C standard */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>

#define sonde_temp "/sys/bus/w1/devices/28-000004efd556/
w1_slave"
#define RELAIS1 0 /*GPIO 17 */

   /* Sous Programmes */
   /* Gestion des Relais */
void relais1(int state)
{
if (state) state=0; else state=1;
digitalWrite (RELAIS1,state);
}

   /* On rappelle que l'on obtient l'information de température
   issue du capteur avec la ligne de commande ci-dessous.
   pi@raspberrypi ~/regul/lecture_temp $ more /sys/bus/
   w1/devices/28-000004efd556/w1_slave
   6c 01 4b 46 7f ff 04 10 5d : crc=5d YES
   6c 01 4b 46 7f ff 04 10 5d t=22750
   pi@raspberrypi ~/regul/lecture_temp $
   */

   /* fonction permettant de récupérer la température en
   degré Celsius de la sonde */

float lecture_DS18B20()
{
FILE *lecture=NULL;
char chaine [255];
char chaine_temp[255];
int cpt,ccpt=0;
float temperature;

lecture=fopen(sonde_temp,»r»);

if (lecture == NULL)
       printf («Erreur a l'ouverture du fichier\n»);
   else {
     fgets (chaine,255,lecture); /* récupération de la ligne :
     6c 01 4b 46 7f ff 04 10 5d : crc=5d YES
     dans la variable chaine */
      fgets (chaine,255,lecture); /* récupération de la ligne :
     6c 01 4b 46 7f ff 04 10 5d t=22750
     dans la variable chaine */
for (cpt=0;chaine[cpt]!='=';cpt++); /* la variable chaine
     contient maintenant
                    la température à partir de = */
       cpt++;
       for (ccpt=0;chaine[cpt]!='\0';chaine_temp[ccpt]=cha
ine[cpt],cpt++,ccpt++);
             {
             sscanf (chaine_temp,»%f»,&temperature);
             }
       fclose(lecture);
   }
return (temperature/1000); /*renvoi la température de la
sonde en degré Celsius */
}

     /* Corps du programme */

int main()
{
const float consigne=20;
float temp;
     /* initialisation de la la fonction wiringPiSetup
      pour utiliser les entrées sorties de la RPi */
wiringPiSetup ();
pinMode (RELAIS1, OUTPUT); /* configuration de ligne
     RELAIS1(GPIO 17) en sortie*/
relais1(1);
float hysteresis=0;

for(;;) /* Boucle infini du process en scrutation constante */

 {
       temp=lecture_DS18B20();
/*      printf(«La temperature est de %f \n»,temp);*/
       if (temp<consigne+hysteresis)
                              {
                              relais1(1);
                              hysteresis=+0.5;
                              }
                    else
                              {
                              relais1(0);
                              hysteresis=-0.5;
                              }
       sleep(5);
     /* permet de libérer du temps (5s)
           pour les autres taches du système */
      }
return EXIT_SUCCESS;
     /* boucle infinie, ce programme ne s'arrête jamais */
     }
```

Le fichier makefile

```
# makefile pour le projet régulation de température.
CC=gcc
CFLAGS=-c -Wall
LDFLAGS=-lwiringPi -g
SOURCES=regul.c
OBJECTS=$(SOURCES:.cpp=.o)
EXECUTABLE=regul

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	→ $(CC) $(LDFLAGS) $(OBJECTS) -o $@

.c.o:
	→ $(CC) $(CFLAGS) $< -o $@
clean:
	→ rm $(EXECUTABLE)
```

Cette ligne de commande ouvre un éditeur de texte dans lequel il faut saisir le fichier « regul.sh » (voir encadré ci-contre).

Pour lancer le « service », il faut saisir la commande suivante :

```
sudo update-rc.d regul.sh start
```

puis

```
sudo service regul.sh start
```

À partir de ces commandes, la fonction de régulation s'exécutera en permanence à chaque mise sous tension du système.

Si toutefois il était nécessaire de stopper le service, saisir la commande :

```
sudo service regul.sh stop
```

Mise en boîte et conclusion

Le prototype complet 11 permet de distinguer les trois parties séparées et isolées électriquement : le RPI, la sonde et la partie puissance comprenant le relais.

Notons que la partie puissance est reliée au réseau électrique domestique 230V AC, il est donc indispensable de respecter la norme NF C 15-100 pour le câblage de cette partie.

Le prototype est ainsi fonctionnel, mais on se rend vite compte que l'on exploite très peu les capacités du RPi. Par exemple, la consigne de température est une constante du programme ; elle n'est donc pas modifiable.

Les étudiants en avance sur l'étude de ce système peuvent imaginer et concevoir des améliorations, comme :

– rendre modifiable la consigne de température en fonction de la date, de la présence des occupants de la pièce… ;

– rendre accessible le paramétrage via Internet ;

Le fichier « regul.sh »

```
#!/bin/bash # interpréteur du script (bash)
case «$1» in
   start)
      # Commandes executées avec le paramètre start
        (celui lors du boot)
      /home/pi/regul/regul &
      ;;
   stop)
      pkill regul
           # Commandes exécutées avec le paramètre stop
             (celui lors de l'arrêt du # processus)
      ;;
   reload|restart)
      $0 stop
      $0 start
      ;;
   *)
      echo "Usage: $0 start|stop|restart|reload"
      exit 1
esac
exit 0
```

– piloter la partie puissance avec une autre loi de commande qui permette une régulation plus fine, par modulation de largeur d'impulsions par exemple ;

– afficher d'autres données du système (température instantanée, consommation, etc.). ■

Pour aller plus loin

Usage du capteur DS18B20
http://www.framboise314.fr/mesure-de-temperature-1-wire-ds18b20-avec-le-raspberry-pi/

Qu'est-ce qu'un module linux
http://doc.ubuntu-fr.org/tutoriel/tout_savoir_sur_les_modules_linux

Regulation
http://fr.wikipedia.org/wiki/R%C3%A9gulation_automatique

WiringPi
http://wiringpi.com

Deamon
http://www.pihomeserver.fr/2013/05/27/raspberry-pi-home-server-lancer-un-programme-automatiquement-au-demarrage/